OSBORNE

Open This Book as Novice and Finish It as a Pro

VISUAL C++ 6 从入门到精通



北京希望电脑公司
北京希望电子出版社

Visual C++ 6 从入门到精通

〔美〕John Paul Mueller 著

希望图书创作室 译

本书配套光盘内容包括：

1. 与本书配套的中文版电子书
2. 送“精通 Visual C++ 5.0”多媒体学习软件

北京希望电脑公司

北京希望电子出版社

1999

内 容 提 要

Visual C++ 6 是 Microsoft 公司最新出品的 C++ 6 开发工具。本书以详实的内容、精选的示例全面介绍了 Visual C++ 6 的新特性以及如何在软件开发过程中运用这些新特性。全书共分五部分：第一部分介绍 Visual C++ 6 的基础知识和基本概念；第二部分介绍数据库编程方法；第三部分阐述 Internet 编程；第四部分说明创建应用程序时应该考虑的安全性问题、普通格式和 HTML 格式帮助文件的创建方法以及应用程序安装盘的建立与分发；第五部分是附录，介绍了 ActiveX 和 OCX 组件资源、在线资源以及本书涉及到的一些技术术语。

本书内容覆盖了开发完善的应用程序所需的各种知识，同时，书中给出了大量与 Visual C++编程相关的站点和讨论组地址，这些地址是真正的 Visual C++ 6程序员上网的站点。本书对已经具有 Visual C++编程经验的读者是很好的升级指导书。对初学 Visual C++的读者来说，利用本书也可以很快入门，迅速达到中级甚至高级 Visual C++程序员的水平。该书对从事 Visual C++ 6 开发与应用的广大科研人员、高校相关专业师生均不失为一本重要的自学、教学参考书。

本书配套光盘的内容是：1.与本书配套的中文版电子书；2.送“精通 Visual C++ 5.0”多媒体学习软件。

需购本书和配套光盘或需技术支持的读者，可直接与 010-62562329，62541992，62531267 或传真至 010-62579874 北京海淀 8721 信箱书刊部（邮编 100080）联系。

版 权 声 明

本书英文版原名为“ Visual C++ 6 from the Ground up **”,由** McGraw Hill **公司出版，版权归** McGraw Hill **公司所有。本书中文简体字版由** McGraw Hill **公司授权出版。未经出版者书面许可，本书的任何部分均不得以任何形式或任何手段复制或传播。**

Visual C++ 6 从入门到精通

〔美〕John Paul Mueller 著

希望图书创作室 译

责任编辑 陆卫民

北京希望电脑公司

北京希望电子出版社 出品

北京海淀路 82 号（100080）

新华书店、新华书店音像发行所发行、各地书店、软件专卖店经销

* * * * *

1999 年 3 月第 1 版 1999 年 3 月第 1 次印刷

开本：787×1092 1/16 印张：33.125

字数：756 千字 印数：1~5000

新出音管[1998] 210 号

ISBN-7-980023-17-X/TP・17

定价：58.00 元（1CD，含配套书）

作 者 简 介

John Mueller 是位自由作家和技术编辑。到现在为止，他已经编著了 40 部著作，撰写了近 200 篇论文。其讨论主题覆盖了从网络到人工智能、从数据库管理到高级编程的广阔领域。Visual Studio 程序员指南和 Windows NT Web 服务器手册是他近期作品中的两部。他在技术编辑方面的经验使他已经帮助了 22 位以上的作者提炼了他们的手稿。除了撰写著作之外，John Mueller 还为 Data Based Advisor 和 Coast Compute 两本杂志做技术编辑服务。作为计算机界认证领域的知名专家，他还在像 Certified Professional Magazine 这样的杂志上发表与认证相关的文章。

当 John 不在计算机旁工作时，你会在他的工作间找到他。他是位干劲十足的木匠和蜡烛制作者，不管在哪天下午，他都会或工作在车床旁，或把做好的物品放置到书架上。他最新的工艺项目之一是制作甘油肥皂，这是制作礼品篮的好材料。读者可以通过 jmueller@mwt.net 在 Internet 上与 John 联系，John 目前正在建立新的 Web 站点：http://www:mwt.net/~jmueller。随意到这些站点看一看、并提些改进建议吧！

Osborne/McGraw-Hill

2600 Tenth Street

Berkeley,California 94710

U.S.A.

致　　谢

感谢我的妻子 Rebecca，是她与我一起完成了本书。如果没有她查找和编辑本书中的某些信息（特别是索引部分），我真不知道是否能够写出这部书。她还详细校对了本书的初稿和终稿。

感谢本书的技术编辑 Greg Guntle，他大大增强了本书所用材料的准确性和深度。

我的代理人 Matt Wagner 保证我得到了第一份合同，并处理了绝大多数作者都没有深思的许多出版细节。

Microsoft 技术支持人员回答了我许多提问，进一步丰富了本书，同时也使我更快地掌握了 Visual C++。与此同时，我还要感谢访问 Internet 各种各样新闻组的人们，他们协助提供了 C++编程技巧的内幕，特别是他们提供了如何使用 OLE-DB 和 ADO 的技巧。

最后，我要感谢 Wendy Rinaldi，Nancy Mclaughlin，Kathryn Hashimoto，以及 Osborne 的全体工作人员，在他们的帮助下本书得以问世。我特别要感谢在事情意外不顺利时 Wendy 所具备的耐心。

译 者 序

Visual C++ 6 是 Microsoft Visual Stutio 套装软件的一个有机组成部分。虽然在构造大多数常规应用程序方面，Visual C++没有市场上常见的快速开发工具（比如 PowerBuilder，Delphi，Visual Basic）的开发效率高，但在速度和代码大小要求较高的场合（比如设备驱动程序等），它依然是首选的开发工具之一。

Visual C++ 6 在以前版本的基础上又增加或增强了许多特性。除了当今的每个开发工具都不可或缺的 Internet 特性外，ADO 数据绑定、ATL 复合控件、AutoCompletion、编辑和继续特性、OLE-DB 提供者模板、延迟加载移入以及新的调试特性等都给 Visual C++ 6 增色不少。

本书作者在熟练使用 Visual C++ 6 的基础上，充分听取各阶层读者对本书先前版本的意见和建议，阅读了 Internet 上有关 Visual C++的大量内容，并与 Microsoft 的技术人员进行了充分交流，最终成就了这部著作。本书分为五个部分：第一部分介绍 Visual C++ 6 的基础知识和基本概念；第二部分介绍数据库编程方法；第三部分阐述 Internet 编程；第四部分说明创建应用程序时应该考虑的安全性问题、两种格式的帮助文件的创建方法以及应用程序的分发；第五部分是附录，介绍了 ActiveX 和 OCX 组件资源、在线资源以及本书涉及到的一些技术术语。

除了详实的技术细节外，本书也为读者提供了大量的与 Visual C++相关的 Internet 站点。通过这些地址，读者不仅能够了解 Visual C++的更多内容及发展趋势，参加一些讨论组提高自己的技术水平，同时也避免了读者在 Internet 上大海捞针般地艰难搜寻 Visual C++

的相关材料。

本书不仅适用于初学 Visual C++的新手，对已经熟练运用 Visual C++以前版本的高手来说，也是一本不可多得的参考资料。

本书由张长富、栾开春、彭涛、张建安、黄石海、姚燕、李匀、黄中敏、陈华等翻译，最后由张长富统稿。参加翻译、录入、校对和资料搜集工作的还有：陈聪、崔立模、宋斌、张剑宇、夏桂娟、王志宏、王锋、张兵、李鹰、王丰、冯桂林、严苏娅、龚楚奇、欧阳志、郝晓伟、苏春波、王刚、李林。翻译过程中得到了享受政府特殊津贴的专家张福林先生的热情指导和帮助，在此深表感谢。由于译者水平所限，欢迎读者批评指正。

张长富

1998,11

引　言

长期使用 Microsoft Visual C++的任何人都会感受到该产品的巨大变化。Microsoft 花了很大的力气来提高 Visual C++这一最新版本的易用性。然而，对 Visual C++程序员来说，编写程序依然是艺术成分多于科学成分。就在 Microsoft 创建新的向导把一个领域的隐秘（比如这个 Visual C++版本中改进的 ATL 向导）暴露于世的同时，某些新技术又把新的隐秘加入到了 Visual C++中（比如 OLE-DB 和 ADO 支持等）。

不断的变化也是你需要另一本 C++参考书的原因。Microsoft 不断地向 Visual C++中增加新特性。虽然本书不能从理论上介绍更多的 C++基本编程知识，但它将会告诉你如何使用 Microsoft 新增的新特性。例如，你会在本书中找到 Visual C++新版本中包含的大量 ActiveX 和其它与 Internet 相关的特性。在 Visual C++的每一次版本更新中，Internet 的主导地位都进一步得到了增强，6.x 版也不例外。正是人们对速度的渴求驱使 Visual C++使用某些 Internet 部件。从 Internet 上下载大型应用程序不是有效的用户选择，用户需要小型、快速的应用程序。听起来很熟悉，是不是？我们将花费大量的篇幅讲述如何把 Visual C++的优势最佳地应用到当今 Internet 驱动的编程环境中。

ActiveX 是用户与分布在 Internet 上的应用程序进行交互的方法之一。它为 Web 网主带来了吸引潜在访问者的更多工具。在许多情况下，你可以得到各种功能而无须编写一行代码。很显然，第一个要问的问题是，“谁关心这些事呢”？答案是想开发内联网或 Internet 站点的所有公司。一项最近的调查表明，当今 48%的公司正在部署 Internet 上的应用，14%

的公司声称他们将在未来的六个月中部署 Internet 应用，28%的公司计划在一年内部署 Internet 应用（总的百分比达到 78%）。显然，人们对 Internet 上的应用兴趣很大。

这个版本的 Visual C++还包括了支持应用程序中嵌入浏览器功能。你可以在多种场合使用浏览器支持功能，包括在你的 Web 服务器上放置帮助文件，以便用户在需要的时候就能访问它们。想一想把桌面帮助浏览器置入应用程序中带来的好处吧。单单支持电话的减少所带来的好处就足以说明应该把浏览器置入应用程序中。这一浏览器支持功能还使旅途中的雇员能够向公司上载数据和从公司下载诸如最新的销售统计以及情况分析之类的信息。

为什么 Internet 会成为这样一种炙手可热的环境呢？另一项调查询问商业客户为什么他们要使用 Internet 提供某种类型的增值服务或为雇员提供更好的信息访问。被调查者中 79%的人认为 Internet 易于使用。访问 Internet 上海量信息所要完成的一切工作就是找个 ISP（Internet 服务提供商）和浏览器。浏览器在安装上并不比任何其它 Windows 应用困难——并且有时候还要容易些。而且，有些 ISP 提供了自动化程度很高的安装程序，用户只需要提供姓名、地址以及诸如信用卡号码这样的付帐信息即可。例如，我最近使用过的 Pacific Bell 安装程序自动创建 Windows 95 下我所需要的连接，然后拨号登录到它们的 Web 站点来帮助我查找最佳的访问号码，并在检查了另一个 Web 站点后完成安装过程。

基于用户使用 Internet 的统计数据，Microsoft 正投资于把分布式数据库支持添加到 Visual C++中。本书中介绍的 OLE-DB 和 ADO 将会告诉你，Microsoft 如何把建立与数据库的连接变得更加容易些，而无论这些数据库放置在什么地方。如果 OLE-DB 连接的提供商允许的话，你甚至可以在 Internet 建立与数据库的连接。

显然，如果说易于使用是人们选择使用 Internet 的原因之一的话，那么许多公司都会有许多使用 Internet 的其它原因。然而，调查中发现使用 Internet 还有其它原因，包括增加

了可用性（69%）、利用了现有基础设施（65%）、成本（51%），以及易于维护（36%）。这项调查允许读者进行多项选择，因此会有上述的百分比。Internet 起到了信息中心仓库的作用，并且或许是绝大多数公司为远程雇员提供访问公司数据库的最佳途径。

任何程序员都知道，代码已经能够运行并不足以说明应用程序已经编写完成。还有许多美学方面的考虑。比如，可以编写出运行速度更快的代码，也可以缩减可执行代码的长度。事实上，编写小型、紧凑代码是专家级程序员的标志。这是个极为重要的目标，贯穿于本书的主题就是如何精炼每一行代码。

编程语言的选择决定了程序员在应用程序中都能完成哪些工作以及以怎样的速度完成这些工作。Visual C++一向以灵活著称：理论上使用 Visual C++可以完成任何任务，并且不会让你的代码变得像意大利式细面条那样乱做一团。C++还以正确使用时能够创建出运行速度极快的应用程序而著称。然而，这些高度的灵活性和快的速度经常给初出茅庐的程序员带来麻烦，也经常形成微妙的故障。许多 Visual C++的反对者也指出使用这种语言时需要很长的开发周期——这是使用诸如 Visual Basic 之类的 RAD（快速应用开发）语言进行编程的用户注意不多的事实。

有些人走得远了些，声称 C++的日子屈指可数。快速的计算机并不像以前那样需要既小且快的代码。而且，随着应用程序越来越复杂，在合理的时间期限内完成应用程序是基本的要求。C++应用程序过长的开发和调试周期确实有点不适应当今的应用程序开发环境。

对 C++程序员来说，幸运的是，有些编程任务 RAD 语言永远也代替不了 C++。即使 RAD 语言已经取代 C++来开发某些类型的应用程序，但对操作系统服务或设备驱动程序来说，程序员依然需要编写简短、紧凑的代码。完成相对底层任务、需要反复执行的代码的编制依然是 C++程序员的工作领域。

本 书 概 要

我已经有点伤害你的好奇心了，现在就让我们看一看本书的主要内容吧。本书分四个重点：通用应用程序编程、与 Internet 相关的编程、数据库编程以及应用程序打包。当然我们也要花点时间讲述编程基础。例如，我们将考察一下最新版的 Visual C++都提供了哪些界面元素以及新的编程特性。

本书的通用应用程序编程一节将向你展示如何使用 MFC AppWizard 创建基本的应用程序。我们不花许多时间来研究创建特定类型应用程序的细节，但将讲述创建 MFC AppWizard 提供的各种类型的应用程序的方法。显然，在本节中我假定读者已经具备了一定的 C++编程经验。当然，我会介绍创建应用程序本身的详细做法，但读者应该知道一些 C++的基本概念，特别是语言元素方面的知识。

数据库编程示例将为已经知道一些数据库基本知识的读者提供丰富的信息。如果你还不知道什么是表、还没有坚实地掌握关系数据库的理论，那么在阅读本书第二部分之前，需要花点时间先熟悉一下上述内容。我们将研究使用 C++操作数据库的基本技巧，你也将理解 ODBC（Open Database Connectivity，开放数据库互连），OLE-DB（Object Linking and embedding database，对象链接与嵌入数据库），ADO（ActiveX Data Object，ActiveX 数据对象）以及 DAO，（Data Access Object，数据访问对象）之间的差别。我们甚至还各花了一章的篇幅来讲述 ODBC 和 ADO 数据库编程。

Internet 编程实际上从本书的第一部分就开始讲述了。这里研究了如何创建基于 HTML 的应用程序。第二部分将讨论如何创建跨越 Internet 的数据库连接，然而这里我们并没有考察任何实际的实例代码。第三部分是介绍 Internet 的中心章节，我们将以中级程序员的水平来研究 ActiveX 编程。如果你只是个初级程序员，那么会发现某些概念有点难以理解。

然而，某些章节，比如第 8 章讲述的 HTML 编程，将会为各层次的程序员提供帮助。另外，第三部分将向读者展示如何使用 ActiveX Document（ActiveX 文档），ISAPI Filter（ISAPI 过滤器），以及 ISAPI Extension（ISAPI 扩展）。

在第四部分，我们将讨论从安全性问题到把帮助文件组合在一起的方方面面的内容。实际上，我们还要探讨 Visual C++中使用帮助文件的两种不同技巧：标准的帮助文件和 Microsoft 正在其产品中使用的新型 HTML 帮助文件。你还将学习发行应用程序的方法，即使在企业环境中这也是一项不可或缺的任务。

通过本书，你将会深入理解 C 编程语言，虽然它还不至于让你成为一名 C 编程高手。本书中你不会找到很多初级的编程知识，但具备一定 C 编程技能的任何人都会理解本书的内容。积极建议读者已经具备 Microsoft Visual C++的一定知识，原因在于所有的示例都以 Visual C++作为基础。即使我们把示例控制在基本编程技巧的范围内，也依然需要编写 OCX 或 DLL 的知识。如果在你阅读本书之前已经掌握了一些底层编程知识，那么你就会很快地学会掌握提供的各种信息。再说一句，本书包含了一些“课程复习”风格的示例，但它们也没有提供很多的初级信息。

当然，上面介绍的仅仅是本书的概貌，在这本书中你会找到能够把 Visual C++的功能发挥得淋漓尽致的各种各样的有用技巧和提示。

本版著作的新内容

本书新内容想法的产生主要有三个来源。首先，我花了大量的时间阅读读者发送来的电子邮件。读者的电子邮件对我编著的每一部书籍都显著地丰富了其内容。第二，Visual C++本身为我提供了一些想法。我必须在本书中覆盖 Visual C++的所有新特性。最后，我花了

大量的时间阅读各种 Visual C++新闻组以查找新的消息线索。某些消息线索十分正规，它们指出了人们使用 Visual C++经常遇到的问题。

第 1 章的绝大部分内容都是全新的。我重新编写了第 1 章，以便提供本版 Visual C++的各种新特性的完整概览。另外，本章的某些部分重新进行了排版，以便能够以更便于阅读的形式向用户提供信息。我竭力以更佳的方式安排插文和表格，让读者能够更快速地找到所需信息。

某些读者抱怨说，在本书的先前版本中我过多地讲述语言的新特性而忽略了普通的编程需要。在本版中，你会发现大大增强了普通应用程序编程材料。第 2 章和第 3 章中提供了几节新材料，并提供了一个新的示例程序。

本书的数据库部分已经进行了几个方面的修订。虽然 DAO 依然是一种可靠的技术，然而它已经过时了。我使用新的 OLE-DB 和 ADO 技术取代了先前书中的 DAO 一节。另外，阐述数据库的章节着重介绍如何通过 Web 服务器连接到数据库上，而不是介绍如何通过传统的网络路由器连接到数据库上。

本书的 Internet 部分也包含了与新技术和改进技术相关的信息。例如，在第 8 章中讨论了 COM+。我还提供了一些 Active Template Library（ATL，活动模板库）的信息。

本书 Internet 部分最大的变化出现在第 13 章。许多读者表示他们希望看到有关 ISAPI 的更详细材料，希望找出如何利用 ISAPI 更有效地使用 Web 服务器。这一章提供了 ISAPI 的更多信息和两个新的编程示例，其中的第二个 ISAPI 过滤器示例显示了如何给你的 Web 服务器增加安全功能。

本书的第五部分也有许多变化，其中最明显的改变是第 15 章讨论的基于 HTML 的帮助。在这里你不仅能够学会如何创建基于 HTML 的帮助，而且也将学会如何把这种帮助添加到你的应用程序中。除了可以提供更佳的帮助信息外，你还会发现安全方面和打包方法

的更新。总而言之，这部分的更新内容可以帮助你向使用你开发的应用程序的人们提供更多、更好的应用程序支持。

系 统 需 求

如果你只是对本书的第一、第二以及第四部分感兴趣，那么阅读本书所需的一切就是一份 Microsoft Visual C++的拷贝。由于我在每个示例中都试图使用 Visual C++的新特性，因此，在绝大多数情况下你都不能使用该产品的老版本（对某些示例来说，Visual C++ 5.0 勉强能用）。无论你使用何种版本的 Windows，一定要确保安装了最新的补丁包和服务包。在测试本书的示例时，我使用的是 Windows 95 的 OSR2 版本和安装了服务包 3 的 Windows NT 4。

如果你打算使用第 14 章的 Internet 信息服务器（Internet Information Server，IIS）示例，那么一定要安装 Windows NT Server 4，并使用服务包 3 进行升级。你还需要在服务器上运行 IIS 4.0。某些示例代码根本不能与以前版本的 IIS 一起工作。第 14 章中的所有示例均假定你使用的是功能完备的 IIS，而不是其它的替代品。

注释 本书中你将学习的许多概念在联机文档中根本就没有。某些概念十分新颖，它们只在一些有选择的 Web 站点上出现。对这样的信息，本书将采用技巧或注释的方式告诉读者它们的地址。另外，有些材料 Microsoft 只通过像 MSDN 订户这样的有选择渠道进行发布。其它一些信息则未被公开，除了在新闻组中某些人偶然发现这些特性外，在其它的任何地方你都找不到它们。

你还需要一台运行 Windows 95/98 或 Windows NT 的计算机作为工作站。你的 Web 服务器和工作站都要求有足够的内存和其它资源以支持本书中用到的各种工具。也就是说，在绝大多数情况下，你的计算机的最低配置应该是 200 MHz 的 Pentium MMX、64MB 内存、至少 3GB 的硬盘空间。虽然你可以降低上述配置，在编写和测试代码时低端计算机会很快让你陷入泥潭之中。我已经在 Pentium 166MHz，32MB 内存的计算机上运行过 Visual C++，但性能让人感到恐怖。

本书使用的约定

本节讨论用法约定。在我们研究匈牙利表示法和如何使用这种表示法时，我们将讨论编程约定。本书使用下述约定：

[<Filename>]	当方括号括住某个值、某个开关或某个命令时，表示该项是个可选项。只有当你想用相应的值、开关或命令提供的功能时，才需要把该项包括到命令行或对话框域中
<Filename>	尖括号中间的变量名是个需要使用实际值代替的名称。你看到的变量名通常是提示词语，你需要提供相应的信息。对左边的示例来说，它表示你需要提供一个文件名。在键入对应值时，千万不要键入尖括号

续表

全部为大写字母	在三种情况下你会看到全部是大写字母：命令、文件名以及大小写敏感的登记项。通常在 DOS 提示符、PIF 文件域、Run 对话框域中键入命令。如果你在其它地方看到了全大写字母，你可以安全地假定它们是大小写敏感的登记项或像文件名这样的其它一些值
File\|Open	菜单和从菜单中进行的选择。“File\|Open”表示“访问 File 菜单，并选择 Open 菜单项”
斜体字	在三种场合你会看到斜体文字：新词、多值项以及不确定的值。无论何时，当实际值未知时，总使用斜体字表示这个值。书中在多个值都可能正确的情况下也使用斜体字印刷，例如，书中你会看到 FILExxxx，它表示相应值可以是 FILE0000 到 FILE9999 之间的任何值
等宽字体文字	将要在宏中输入的文本或在命令行上键入的文本与解释它们的文字区分开来是相当重要的，本书使用等宽字体来区分两者。当你看到等宽字体的文字时，就表示这些信息要出现在宏中，包含在像 CONFIG.SYS 或 AUTOEXEC.BAT 这样的系统文件中或者是需要在命令行上键入的文本。你甚至会看到 Windows 命令中的开关也使用这种风格的文本。还有一种情况使用等宽字体格式的文本。每一个代码列表均使用等宽字体，这样更易于阅读。使用等宽风格的文本还使得易于在文本中增加某些东西，比如代码示例中的缩进

标志

本书采用了许多帮助读者区分不同类型信息的标志。下文描述了各标志的用途。

注释 注释告诉读者一些有趣的事实，它们并不影响读者使用本书的其它信息。我使用注释向读者提供一些在使用 Visual C++、Windows NT 或 Windows 95 时收集到的一些有用信息。

技巧 每个人都喜欢技巧，原因在于它们会告诉读者一些完成某项任务时以前或许从未想到的方法。技巧框还提供了完成某项任务的另一条途径，读者或许会认为这一条途径比正文中提到的方法更好。

警告 它表示你要留神了。警告几乎总是要告诉读者，如果你要完成某项操作（或不能完成其它操作）时，将会引发某些系统破坏或数据损坏。在你执行警告词语之后的任何指令之前，一定要确保彻底地理解了这个警告。

Web 链接 Internet 包含了丰富的信息，但最低限度上说，找到这些信息是比较困难的。Web 链接帮助读者在 Internet 上找到能够提高读者编程能力或能够学到新的编程技巧的新的信息资源。读者还会发现一些新闻组的 Web 链接，它告诉你在什么地方可以找到谈论 Visual C++的其他人群。最后，Web 链接还会帮助读者找到一些能够使编程工作比以前更快、更容易的实用程序。

可移植性 保证代码能够在各种情况下和各种平台上运行总是个美好的理想。当你看到这个标志时，它提醒你一些把代码或可执行文件从一台机器上迁移到另一台机器上时可能存在的问题。如果你的代码要在多台不同类型的机器上运行或要在多种操作系统环境中运行的能力对你来说十分重要的话，一定要特别留意这种类型的标志。

高级技巧 高级技巧框中包含了一些额外的信息，即使你不知道这些信息，也并不影响你编写出优质的程序。高级技巧框为中级和高级程序员提供了一些有趣的资料，他们可以使用这些资料来增强其应用程序的功能或提高自己的生产效率。

本书中读者还会看到各种旁注。它们描述了在你开始某个过程或完成其它任务前应该记住的一段信息。旁注还包含了一些诸如文件位置或应该查看某个示例程序的某些内容这样有用的琐碎信息。在绝大多数情况下，旁注是简单实用的信息宝藏，你可以利用它提高你的整体编程水平。

匈牙利表示法概述

密码——间谍电影和其它各种人类活动的基本素材。当你第一次看到匈牙利表示法时，你可能会把它当成另一种密码。该表示法包含了密码的一切要素，其中包括一系列不得不解码的神秘字符以及使用时几乎不能破译的结果。然而，不久你就会明白，这是其他程序员的秘密代码，而不是本书中使用的匈牙利表示法。

匈牙利表示法可以为你节省大量的时间和精力。在编程方面花过大量时间的任何人都

知道，当阅读以前自己编写的代码或阅读其他人编写的代码时，好的文档是无价之宝。这也正是匈牙利表示法要为你完成的任务——文档化代码。

对匈牙利表示法的清晰理解将有助于你从本书的示例以及 Microsoft（以及其他厂商）的手册中学到更多深入的东西。每一个 Windows 编程语言厂商都在其手册中使用某种形式的匈牙利表示法。另外，相同的概念同样适用于诸如 Visual FoxPro，Delphi 以及 Visual Basic 这样的编程语言。即使语言本身完全不同，但使用匈牙利表示法编写出的代码超越了编程语言而具备一定的相似性。

那么，严格讲起来，什么是匈牙利表示法呢？匈牙利表示法是一种告诉其他人你准备如何使用变量的表示方法。知道了变量要干些什么经常有助于解释代码本身。例如，如果我告诉你某个特定变量保存了某个窗口的句柄，那么这就比变量只简单地是个变量提供了更多的信息。理解了该变量将操作窗口后，你就可以解释代码了。

这种变量命名系统第一阶段的开发工作是由 Microsoft 公司的 Charles Simonyi 完成的。他把这种命名系统称做匈牙利表示法（Hungarian Notation），我们在这里就使用这种称呼。你可以从许多地方获取 Charles Simonyi 著作的副本，包括 BBS 和 Internet 上的某些 Microsoft 编程站点（像 CompuServer 这样的许多联机服务也以各种形式提供匈牙利表示法的副本）。其他开发人员对 Simonyi 的研究成果做了进一步的增强。例如，Xbase 程序员使用他们自己特殊版本的匈牙利表示法。这种表示法表达了 Xbase 提供的不同类型的变量。Fresh Technologies 的 Robert A. Difalco 出版了匈牙利表示法的 Xbase 增强版本。在某些针对 DBMS 的 BBS 上以及 CompuServer 的 Computer Associates Clipper 论坛上你都可以找到这位作者的作品。

本节讲述的基本概念在前面提到的两个文档中你都可以查阅到，表达形式上可能会有所不同。本节旧话重提的目的是为了让你准确地理解我所采用的约定的意义，并说明在你

的代码中应该如何最佳地使用它们。在你的代码中采用这种命名法的原因有四个：

帮助记忆值　这种表示法有助于更容易地记住变量的名称，在团队项目中这是个要认真考虑的重要事情。

提供建议值　你或许并不是唯一要修改你的代码的人。如果你正在开发一个团队项目的话，小组中的其他成员最起码也要看一看你编写的代码。使用这些约定有助于其他开发人员理解你的代码。

一致性　程序员的工作成果通常并不仅仅体现在效率和功能方面，而且也体现在编写出的代码是否能够被其他程序员轻易地读懂。使用这些约定有助于在不同的项目中保持一致的代码风格。利用你所使用的约定，其他程序员也可以轻易地加入到修改或编写代码的行列中。

加快判断速度　在商业化世界中，创建和修改代码的速度经常会决定特定项目的成功程度。使用一致的代码将减少你花在猜测其他人创建的变量或函数意义方面所需的时间。这一判断时间的减少也就增加了你用于有效地开发产品的时间。

上面已经讲述了为什么应该使用匈牙利表示法，现在让我们看一看本书中准备如何应用这一表示法吧。我将按照下节阐述的规则命名变量。在命名数据库字段或其它与值相关的结构时我也将使用这些规则。另外，只要匈牙利表示法有助于更清晰地表达函数和过程的意义，这些函数和过程的命名也将使用下节中介绍的规则。

规则 1：使用变量前缀

变量命名时，总使用一个或两个小写字母作为变量名前缀，以指明变量的类型。绝大多数情况下，变量名前缀使用变量类型的第一个字母，因此，可以轻易地记住应该使用哪个字母。下面的示例显示了 Visual Basic，Delphi 以及 C 语言中普遍使用的前缀字母(Windows

中有成千上万种文字组合没有在这里列出）。下表也提供了几个数据库专用的修饰符：

a	Array
c	Character
d	Date
dbl	Double
dc	Device Context
dw	Double Word
f	Flag,Boolean **或** Logical
h	Handle
i	Integer
inst	Instance
l	Long
li	Long Integer
lp	Long Point
msg	Message
n	Numeric
o	Object
pal	Palette
psz	**指向以零结尾的字符串的指针（**Pointer**）**
ptr	**指针（与其它类型变量一起使用时也可以使用** p**，比如** psz**）**
r	Real

续表

rc	Rectangle
rgb	Red,Green,Blue（红、绿、蓝，颜色变量）
rsrc	Resource
sgl	Single
si	Short Integer
sz	Zero Terminated String
u	Unsigned
ui	Unsigned Integer 或 Byte
w	Word
wnd	Window

规则 2：标识状态变量

某些变量用于指明像数据库、字段或控件这样的对象的状态。这些变量甚至可以保存其它变量的状态。告诉其他程序员某个变量用于监视某个对象的当前状态有助于他们理解这些变量在程序中的意义。你可以使用下面的三字母修饰符之一来指明该变量是状态变量：

New	新的状态
Sav	已保存状态
Tem	临时状态

规则 3：应用标准修饰符

标准修饰符几乎可以让其他人立即看清楚变量的用途。标准修饰符不提供变量的类型

信息，但它们说明了一个变量与其它变量之间的关系。例如，使用 Clr 修饰符将会告诉阅读程序的人该变量以某种方式用于操作颜色。你甚至可以把多个修饰符组合起来以进一步说明变量的作用并描述清楚应该如何使用这个变量。例如，cClrCrs 是个字符型变量，它用于确定光标的显示颜色。使用一到三个这样的修饰符通常就足以描述清楚变量的用途了。下面列出的修饰符是应用最普遍的一些修饰符：

Ar	Array
Attr	Attribute
B	Bottom
Clr	Color
Col	Column
Crs	Cursor
Dbf	Database File（数据库文件）
F	First
File	File
Fld	Field
L	Last/Left
Msg	Message
Name	Name
Ntx	Index File（索引文件）
R	right
Rec	Record Number（记录号）

续表

Ret	Return Value（返回值）
Scr	Screen
Str	String
T	Top
X	Row
Y	Column

规则 4：添加描述性文字

当你清楚地定义了变量的内容和用途后，就可以使用一些描述性的文字进一步精化这些定义。例如，你可以把指向用于保存雇员姓名的字符串的长型指针变量定义为：lpszEmpName。这个变量的前两个字母告诉我们它是个长型指针，紧接着的两个字母告诉我们这是个以零（或空）结尾的字符串，其余的字母告诉我们它是个雇员姓名（请注意，在这个示例中我使用了标准修饰符 Name）。在代码中看到这样的变量名让人一眼就可以看出其类型和用途。

规则 5：创建多个变量

时不时就会遇到在某个特定模块中使用一个变量并不能够满足每一种需要的情况。多数情况下你会想到创建同种类型的多个变量，并用数字来区分这些变量。当然你也可以使用下面的这种有意义的数字指示符来说明各变量的功能：

1，2，3	**状态指针引用，比如** cSavClr1，cSavClr2，…
Max	**限制上界，比如** nFldMax **指明字段的最大个数**
Min	**限制下界，比如** nRecMin **指明记录的最小个数**
Ord	**某种类型的顺序号**

目　　录

引言

本书概要

本版著作新内容

系统需求

本书使用的约定

第 一 部 分

第 二 部 分

第 三 部 分

第 四 部 分

第 五 部 分

第一部分

Visual C++基础



返回总目录

目　　录

第 1 章　Visual C++引论 3

1.1　为什么要用 Visual C++ 5

1.2　本版 Visual C++的新特性 6

1.3　一 些 缺 陷 23

1.4　考察界面元素 25

第 2 章　建立基本应用程序 41

2.1　了解应用程序类型 42

2.2　编写控制台应用程序 55

2.3　编写基于对话框的应用程序 66

2.4　编写单文档应用程序 82

2.5　编写基于 HTML 文档的应用程序 92

第 3 章　理解 Visual C++的资源 100

3.1　定制使用应用程序向导生成的应用程序所用的资源 103

3.2　使用加速键和菜单 129

3.3　使用工具条 136

第1章　Visual C++引论

只有真正的程序员才使用 Visual C++。反过来，则并不见得如此。最近似乎有许多专业的程序员放弃了使用这一语言而改用了其它产品提供的 RAD（快速应用开发）环境。许多人认为 Visual C++太古老、太烦琐、学起来太困难了。

说句实在话，Visual C++的学习周期确实比其它语言要长。可以很轻易地引入需要花上数小时才能排除的微妙错误这一事实一点也无助于提高程序员的开发效率。

然而，让我们先把这些负面缺陷暂时放到一边，来看一看 Visual C++具备的长处。使用 Visual C++主要的原因之一就在于它的灵活性。你可以彻底地控制整个开发环境。其它语言则更多地趋向于庇护程序员，当要做一些基本的东西时，它们会做得很好。不幸的是，当你需要使用像 C++这样的语言删除编程中的繁文缛节，并把任务完成时，这种保护作用就变成了开发工作的障碍。

长期以来，Visual C++一直拥有能够创建短小高效程序的美誉。使用这种语言编写的程序几乎可以与用汇编语言编写的程序达到相同的运行速度，并且避免了汇编语言存在的各种问题。C++实际上是介于汇编语言中寄存器编程的神秘莫测和像 Pascal 这种保护型编程环境方便性之中间难度的语言。

C++是编写诸如操作系统、设备驱动程序以及动态链接库（DLL）的强大语言，这一点不会不引起你的注意，这三种领域代码的开发依然是 Visual C++的主战场。Visual C++生成的短小、快速的代码在操作系统类对时间要求很高的

系统中获得了极高的赞誉。

到现在为止，我告诉你的一切你可能已经在其它书籍中看过无数遍了。下面就让我讲一些这个版本的 Visual C++中独有的内容。Microsoft 添加到这个版本的 Visual C++中的最新特性之一是更佳的原型能力，这一点通过增强的向导来实现。现在，这个特性还不能把 Visual C++提升到像 Visual Basic 向导相同的水平上，但它确实减少了开发应用程序的入门时间——这是个深受欢迎的变化。另外，增强的向导使得这个版本的 Visual C++比以前版本更为友好（本章的后续部分我将全部用于介绍 Visual C++的其它新特性）。

Visual C++也是编写 ActiveX 控件以及像 ISAPI（Internet 服务器应用程序接口）扩展和 ISAPI 过滤器这样针对 IIS 的专用代码的理想编程环境。即使对属于快速应用程序开发环境中的应用程序编程来说，也没有人愿意花时间从Internet 上下载一个巨型控件，也同样没有人愿意让慢速的过滤器加重 Web 服务器的负担。Visual C++可以生成人们真正想要的简短可执行文件。另外，Visual C++提供的额外灵活性也使得编写这些类型的应用程序更加容易些。本书中我们将花些时间来研究 ActiveX 控件，因此这里我就不再多讲了。我还为使用 Web 服务器的人们准备了几个 ISAPI 扩展和 ISAPI 过滤器的示例。

Visual C++擅长的另一领域是数据库编程。我并不是说要使用 Visual C++编写一个功能完备的仓储控制系统——那样确实要花去很长的时间。然而，Internet 用户也确实需要访问数据库管理者那里的数据。Visual C++能够在不比 RAD 语言多多少编程的情况下提供快速的数据库访问。

1.1 为什么要用 Visual C++

我们已经研究了作为编程语言为什么要使用 C++的一般性问题。然而，对你想生成 Windows 应用程序时为什么要使用 Visual C++却谈论得不多。最基本的原因很好理解：由于 Microsoft 是 Windows 的生产厂商，那里的人们当然最清楚地知道该操作系统内部的东西是如何运行的。在 MFC（Microsoft 基础类库）的 Windows API 钩子中你会找到这些内幕知识。

Visual C++还提供了许多在其它产品中或许找不到的特性。尽管你可以通过构建自己的工具来克服其它产品存在的任何缺点，但为什么非要从头造车轮呢？本书中我们讨论最多的两个特性是数据库编程和 ActiveX 编程，两者都包括控件和文档。我们还将讲述与 Internet 相关的主题——比如 ISAPI——以及新的应用类型，比如基于 HTML 的应用。Visual C++让你能够在一个产品中访问所有这些技术，而无须购买第三方的产品。

持久性是使用 Visual C++的另一个原因。与我聊过天的一些程序员关心计算机界某些公司的未来。他们为这些公司以后不能对其产品提供支持而深感担忧。开发人员不仅今天需要、而且未来也需要得到强有力的支持。由于 C++的每种实现都存在一些细微的差别，特别是在你使用厂商提供的特有特性的情况下，你的应用程序确实与开发工具产品的命运联系在一起了。我们可以负责地说，在不远的将来 Microsoft 不会跑到哪儿去——使用它的产品就意味着你总可以升级到更好的产品，当然要在这些产品发行的时候。

1.2 本版Visual C++的新特性

近来软件开发界流传的一个词是 Internet。你现在见到的每个产品都具备某些与 Internet 相关的小玩意儿，Visual C++也不例外。在本书中我们将考察与 Internet 相关的新特性和升级特性。表 1.1 列出了 Visual C++新特性的完整列表，并显示了哪些版本包含这些新特性（显然，企业版包含的特性最完整）。在随后的段落中你会找到 Visual C++ Internet 特有特性的概览，从而决定 Visual C++ 如何最佳地来满足你的 Internet、桌面以及数据库编程的需要。

表 1.1 Visual C++版本之间特性比较

特性	学习版（标准版）	专业版	企业版
活动模板库（ATL）	X[1]	X	X
应用分发许可	X	X	X
AppWizard 和数据源	X	X	X
通过 OLE-DB 进行的 ASA400 数据库访问			X
AutoCompletion（自动完成语句）	X	X	X
客户/服务器应用开发	\[2]	\	X

续表

代码优化和剖析器		X	X
创建定制化应用向导		X	X
数据绑定控件（RemoteData）		X	X
扩展的存储过程向导			X
InstallShield		X	X
Internet 信息服务器（IIS）4.0			X
MFC 数据绑定		X	X
MFC 数据库类（DAO，ODBC 以及文件 I/O）	X	X	X
MFC 数据库类（OLE-DB 和 ADO）；也称做 OLE-DB 模板		X	X
Microsoft Transaction Server (MTS)			X
代理服务器			X
远程自动化部件			X
远程数据对象（RDO）		X	X
SNA 服务器			X
SQL 数据库工具		X	X
SQL 编辑、调试以及存储过程调试			X

续表

SQL Server 6.5 Developer 版和服务包 3			X
静态链接 MFC 库		X	X
系统管理服务器（SMS）			X
TSQL 调试			X
可视化数据库工具		\	X
可视化建模器			X
Visual SourceSafe 版本控制			X
Visual Studio Enterprise Edition（VSEF）解决方案教程			X

[1]X 完全兼容

[2]\ 部分兼容

/GZ 编译器选项（在调试模式下捕获发行版本中的错误）

你曾经遇到过在你的发行版中碰到了运行错误，但在调试模式错误又消失得无影无踪的情况吗？/GZ 编译器选项正是用于帮助你排除这类错误的法宝。该编译选项允许你使用/O1，/O2，/Ox 或/Og 优化开关，从而找出发行版错误的根源。然而，这时 Visual C++依然对代码中任何以#pragma 开始的语句进行优化，这就是说，有可能不会找出所有的优化缺陷。除此之外，/GZ 编译器选项还将完成下述三项任务：

自动初始化局部变量 Visual C++将自动把局部变量初始化为值 OxCC，这样你就会更快地找出未初始化的变量。该选项还会以 C4700

，这样你就会更快地找出未初始化的变量。该选项还会以 C4700 或 C4701 警告信息让你找到未正常引用的变量。

函数指针调用栈有效性检查　Visual C++检查每个函数，确保函数正常地使用堆栈。函数退出时检查 ESP 寄存器的值，以确保与函数进入时的值相同。这种检查让你能够发现，在使用函数指针调用函数时被调用函数期待的清理堆栈方式与调用函数提供的清理堆栈方式之间的不匹配情况。

调用堆栈有效性检查　Visual C++在每个调用结束时检查堆栈指针（ESP 寄存器），以保证该指针没有发生变化。这种检查让你能够发现内联汇编例程或函数调用中不匹配的错误。

_ _强制内联关键字

强制内联关键字在 Visual C++的前一个版本中已经提供，在这个版本中功能又进一步得到了增强。通过把被调用函数直接展开到调用函数内部的方法，内联函数避免了函数调用的消耗。使用内联函数可以加快应用程序的运行速度，然而不幸的是，内联代码也增加了可执行代码的长度，原因在于每当内联函数被调用时，都要嵌入到调用函数中。

正常情况下，Visual C++要对内联函数进行成本分析，以确定使用内联方式增加的代码长度是否能够得到相应的速度优势。在两种情况下，Visual C++按正常方式创建内联函数的独立副本并以函数方式而不是内联方式调用它：递归函数调用和在转换单元其它地方引用的函数。

_ _强制内联关键字通知 Visual C++不执行正常的成本分析并且总把内联函数编译成内联代码，换句话说，程序员负责考虑某些情况下（比如递归函数调用）可能产生极大规模代码的问题。因此，在使用这个新的关键字时要十分小心，确保代码的规模保持在合理的范围内。

即使使用了_ _强制内联关键字，在某些情况下编译程序依然不生成内联代码。例如，使用/Ob0 命令行开关（调试模式的一个开关）就不生成内联代码。不能生成内联代码的另一种情况是被调用函数使用了变长参数表。无论什么情况下，当编译器不能生成内联代码时，它都会向你发出第一级的警告信息（编号为 4714）。

ADO 数据绑定

Microsoft 不断地努力改进其编程语言产品的数据访问能力。ADO（ActiveX Data Object，ActiveX 数据对象）代表了提供数据访问的一条新的途径，它通过把数据绑定 ActiveX 控件与 ADODC（ADO Data Control，ADO 数据控件）结合起来来实现。ADODC 作为数据源定义了要显示信息的存储位置和数据访问需求。需要为 ADODC 提供六段信息：OLE-DB 提供者的名称（比如 SQL Server）、DSN（像在控制面板的 ODBC 设置中那样指定的数据源名称）、用户名、保密字、记录源（通常是个 SQL 查询）以及连接字符串。ActiveX 控件用于显示数据源的内容。

与以前的数据库访问方法相比，ADO 有数项优点。下面就为你介绍这些优点：

独立地创建对象　你不再需要必须遵循对象的层次。这一特性允许你只创建所需要的对象，从而减少了内存需求并加快了应用程序的运行速度。

成批更新　代替把每次更新都发送给服务器，你可以在本地内存中收集这些更新然后一次将所有更新都发送给服务器。这样我们就改进了应用程序的性能（原因在于可以在后台进行更新），并降低了网络的负载。

存储过程　存储过程作为数据库管理器的一部分驻留在服务器上，并且能够在指定数据集上完成特定的任务。ADO 支持使用带有输入/输出参数和返回值的存储过程。

多种游标类型　从本质上讲，游标指向当前要操作的数据。理论上说，你甚至可以使用特定的后台游标。

返回行数限制　你只得到能够满足用户实际数据需求的数据量。

多记录集对象　该功能支持操作存储过程或批处理过程返回的多个记录集。

自由线索化对象　增强了 Web 服务器的功能。

有两种数据绑定模型可用于 ActiveX 控件。第一个模型是简单的数据绑定，它允许使用像文本框这样的 ActiveX 控件显示一条记录的一个字段；第二个模型是复杂的数据绑定，它允许使用像网格这样的 ActiveX 控件同时显示多条记录的多个字段。复杂的数据绑定也要求 ActiveX 控件管理要显示哪些记录和哪些字段，这是简单数据绑定时 ADODC 通常要完成的任务。

Visual C++本身带了一些支持 ADO 的 ActiveX 控件，其中包括：

DataGrid

DataCombo

DataList

Hierarchical Flex Grid

Date and Time Picker

ATL 复合控件

这一特性让你能够按自己的方式使用其它控件创建出新的 ATL（Active Template Library，活动模板库）控件。定制这样的控件时要打开一个对话框，然后在对话框中填入窗口和 ActiveX 控件。该对话框占据了新控件的整个客户区。使用这两个向导之一能够创建 ATL 复合控件：ATL COM AppWizard 或 ATL Object Wizard。

绝大多数程序员使用这项技术的原因是它能够创建一个作为单一 ActiveX 控件显示的窗体。例如，如果你现在正要求用户下载一个有许多控件的窗体来显示 Web 页面，那么使用 ATL 复合控件可以减低下载任务的复杂性。当你修改了 ATL 复合控件时，用户只需要下载一个控件也就可以了。这种下载方式不仅加快了控件的下载速度，而且也降低了下载失败的可能性（要么下载整个控件，要么不下载——而不会下载部分窗体）。

AutoCompletion

AutoCompletion 特性帮助你完成 C++头文件和源代码文件中的语句。应用该功能时，你会看到一个带有完成代码行可能选项的列表框。例如，考虑下面的代码：

```
CString  oMyString;

oMyString.
```

当你在 oMyString 后输入圆点时，你就会看到一个包含了 CString 对象的各种方法和属性的列表框，此时，你可以向下滚动列表框，找到所需的属性或方法后单击该属性或方法，相应的属性或方法就会填充到原点后面。当然也可以输入你要使用的方法或属性的前几个字母，列表框会随着输入过程的进行而朝希望的名称滚动。一旦你选择了某个方法或属性，Visual C++将自动键入你要使用的属性或方法的名称。这样就大大减少了文字输入的时间，从而有更多的时间编写代码。

技巧 如果你的环境中丢失了 AutoCompletion 列表框，那么在该列表框先前出现过的地方右击，然后从上下文菜单中选择 List Members 菜单项，这个列表框就会重新显示在你的面前。

现在假定你正在查看 AutoCompletion 列表框，但还不能肯定 CString 对象的 AnsiToOem()方法存在的限制。如果你加亮这个列表项并把鼠标指针放置到这个列表项上，就会看到一个气球帮助，它告诉你该方法最多可处理 255 个字符。

这种气球型帮助正是 Visual C++的另一个新特性，称做文档注释（Doc Comment）。

文档注释并不仅仅是 Visual C++软件包的一部分，你也可以把这样的注释添加到你的函数和变量上。方法很简单，你所要做的就是或者在函数或变量说明的前面加上一条注释，或者把注释放置到变量或函数说明的同一行上。利用这个特性你可以向其他人提供函数或变量的各种信息。显然，你不会希望把这个注释弄得很大，Microsoft 把文档注释的长度限制为 12 行，对绝大多数情况来说在这个限制下也足够使用了。

文档注释并不局限于在 AutoCompletion 列表框中使用。把鼠标指针放置到某个变量或函数上，你会看到在与 AutoCompletion 列表框中显示的相同的文档注释气球。正如你所看到的，文档注释是为你的应用程序提供连续文档的迅捷方法，并且也使得阅读你的代码的其他人更方便地理解代码。

改进的 ClassView 和 WizardBar

Microsoft 为 ClassView 和 WizardBar 都提供了一些新特性。下面的列表予以介绍：

删除成员函数　现在不需要你做许多工作就可以删除成员函数了。你所必须完成的所有任务就是把光标放置到成员函数内部，然后从 WizardBar Action 下拉列表框中选择 Delete 或在 ClassWizard 对话框的 Message Maps 选项卡中选择 Delete Function。无论使用哪种方法，Visual C++都从头文件中删除函数声明并在源代码中注解掉函数体。这种处理方式让你在需

要被删除掉的函数时，可以轻易地恢复该函数。

转移到对话框编辑器　这个功能只在创建对话框风格的应用程序时可以使用。选择包含所需对话框的类，然后在 WizardBar Action 下拉列表框中选择 Go to Dialog Editor 选项。

新窗体　向你的工程中添加新的窗体。你会看到一个 New Form 对话框，在这个对话框中选择窗体名、窗体的基类（最常用的基类包括CdaoRecordView，CformView，CDialog 以及 CRecordView）、以及资源编号。基类的选择部分依赖于你正创建的应用程序的类型和使用应用程序向导过程中所做的各种选择。某些情况下，你也将可以选择自动化方式并提供文档的模板信息。当某些选项当前不能使用时，Visual C++会把这些选项变成灰色方式显示。

跟踪空类　现在 WizardBar 可以跟踪所有类，即使那些没有成员函数的类也一样能够跟踪。如果你选择了某个没有成员函数的类，那么WizardBar 的 Members 组合框中将显示“ No Members - Create New Class ”。缺省的 WizardBar 动作是向空类中添加一个新的成员。

注　使用 Insert | New Form 命令也可以打开 New Form 对话框。

命令行参数

Visual C++提供了一个崭新的 MSDEV 实用程序，利用这个实用程序，你可以直接从命令行来编译工程，而无须首先移出 MAKE 文件、然后再使用 NMAKE 实用程序来完成编译。MSDEV 实用程序的语法为：

MSDEV <Filename> [/MAKE "<Project Name> - <Configuration Name> |

ALL"] [/REBUILD /CLEAN /NORECURSE /OUT <Log File> /USEENV]

你必须提供 DSP**（工程）或** DSW**（工作区）文件名。下面的文字描述了可选的命令行开关：**

/MAKE **该选项让指定编译时使用的工程和配置。工程名（**Project Name**）总是工程创建过程中你为工程起的名称。配置名（**Configuration Name**）既可以是像** Win32 Debug **这样的特定名称，也可以是所有可用的配置（使用** ALL **关键字）。**Visual C++**支持在一行上使用多个**/MAKE **参数，因此，你可以创建任意多个配置。**

/REBUILD **在开始编译前首先清除所有中间文件，然后建立新的中间文件。这种方式保证进行完整编译。**

/CLEAN **在开始编译前首先清除所有中间文件，但并不重建中间文件。**

/NORECURSE **编译当前工程，但并不编译与该工程相关的其他工程。**

/OUT <Log File> **把屏幕输出重定向到一个记录文件中。利用该记录文件可以分析编译中发现的错误并进行相应校正。**

/USEENV **忽略** DSW **或** DSP **文件中的** tools.options.directories **设置，并用当前环境中的设置进行替代。这样就可以不用重新打开** Visual C++**而覆盖环境设置，但这种做法也意味着可能会产生错误。**

全面改进的编译器

编译大型 PCH 文件的工程将花费更少的时间。虽然你会注意到对小型工程速度的改进，但对使用大型 PCH 文件的工程来说，这种改进更加明显。

延迟加载移入

正常情况下，有两种方式使用 DLL：可以让应用程序自动加载 DLL，也可以根据需要手工地加载和卸载 DLL。延迟加载特性为你提供了第三种选择，它允许应用程序在需要时自动加载 DLL。这就是说，如果用户未要求使用某种服务——比如说你的字处理应用程序中的拼写检查——那么系统就不会加载相应的 DLL。

对程序员来说，使用这一特性并不需要做多少工作。你要完成的所有任务就是在链接应用程序时加上/DELAYLOAD 命令行开关并把 DELAYIMP.LIB 添加到链接文件列表中。

动态 HTML

新的 MFC 类 CHTML 允许你把动态 HTML（DHTML）视图添加到你的应用程序中。这个类真正的功能就是在不需要多做多少编程的情况下可以把 Web 浏览器加入到应用程序中。显然，由于 Visual C++是微软的产品，你的应用程序中的 Web 浏览器将是 Internet Explorer 4 特性的翻版。这个特性让你能够以对用户近乎透明的方式把桌面应用和 Internet 应用混合在一起。

动态语法分析

随着向应用程序中键入新的类、函数以及变量信息，ClassView 现在能够自动进行更新。这样我们随时都可以看到应用程序中的变化而无须等待将它们保存到源文件中。这个特性也意味着像 WizardBar 和 ClassWizard 这样的部件总是最新的。

注释 这个特性现在还存在一些问题，微软许诺在该产品发行时这些问题将得到纠正。例如，某些输入错误将会导致 Visual C++不正确地猜测你的意图，结果造成了 IDE 的死循环。在所有情况下，重新启动 IDE 就会清除这样的问题。

编辑和继续特性

编辑和继续特性让你能够在调试应用程序时对应用程序做些小的改动，而无须从头重建工程。这个特性不仅让我们能够进行更快速的调试，而且也可以对应用程序执行“如果——那么”类型的分析。

使用编辑和继续特性时也存在一些限制。如果修改的代码超过了编辑和继续特性能够完成的范围，那么 Visual C++简单地提示你重建工程。下面的列表给出了不能使用编辑和继续特性的一般原则：

- 头文件
- 全局/静态数据
- 类定义和函数原型

- 引入新的变量类型
- 例外处理模块

编辑和继续特性在缺省情况下是打开的，因此使用该特性时不需要做任何特殊的工作。如果想关闭该特性，那么使用 Tools | Options 命令来显示 Options 对话框，选择 Debug 选项卡，不选中 Debug Commands Invoke Edit and Continue Debugging 选项。当打开或关闭该特性后，需要完整地重建工程。

扩展存储过程向导

存储过程让你能够以某种方式操作数据库中的数据并把结果输出给用户。Visual C++提供了一个新的向导帮助你创建 SQL Server 的存储过程。这些存储过程使用了带有名称为 Extended stored procedure（扩展存储过程）的函数的 COM 接口。当我们在 SQL Server 上注册了扩展存储过程 DLL 后，就可以访问使用标准 COM 接口检索出的数据了。

新的调试特性

Visual C++提供了丰富的新调试特性，这些特性让你能够更容易地找出应用程序中存在的缺陷。表 1.2 列出了所有这些新特性，并给出了每个特性的简单描述。

表 1.2　新调试特性概览

特性	描述
VARIANT 和 GUID 的自动扩展	AUTOEXP.DAT（自动扩展）文件包含了以更有意义方式显示 VARIANT 和 GUID 的新规则（AUTOEXP.DAT 文件放置在硬盘的 Program Files\Microsoft Visual Studio\Common\MSDev98\Bin 目录中）。例如，VARIANT 现在将以实际包含数据的数据类型方式显示；字符串就显示为字符串。另外，Visual C++同时显示 VARIANT 类型以及相应的值。GUID 并不一定要显示为一串数字，Visual C++将努力在注册表中查找 GUID 的名称并以名称进行显示
调试器格式化符号	Visual C++支持完整的新调试器符号： ,hr 像通用 COM 返回值那样显示 32 位结果或出错代码，比如 S_OK 或 E_NOTIMPL。如果它不能显示精确的出错值，Visual C++将试图把返回值转换成像“Not Enough memory”这样的出错信息。
	,mq 让你以四个四字方式显示内存，以此代替 64 位寄存器。
	,st 根据 AUTOEXP.DAT 文件中 Unicode Strings 设置值决定以 Unicode 还是 ANSI 格式来显示字符串。
	,wc 以解码的 Windows 类标志（WC_constants）来显示数值值。

续表

	,wm 以解码的 Windows 消息值（WM_constants）来显示数值值	
反汇编输出	在反汇编输出中提供了更多的信息，其中包括可用时的符号信息	
Load COFF & Exports	该选项放置在 Options 对话框的 Debug 选项卡中（使用 Tools	Options 命令可以打开该对话框）。这个选项让你加载附加的调试信息，这些信息可以被不是 Visual C++的其它编译器使用（Visual C++使用 CodeView 格式的调试信息）。例如，Visual Basic 在 MSVBVM50.DBG 文件中提供了通用对象文件格式（COFF）的调试信息。当没有其它调试信息（至少是 Visual C++能够识别的信息）可用时，移出选项就发挥作用了。Visual C++将加载每个 DLL 移出表的内容并将其转换成用于调试目的的符号信息
寄存器	几个新的寄存器将帮助我们找出应用程序中的更详细信息。Thread Information Blocks（TIBs，线程信息块）伪寄存器提供了当前线程的信息。ERR 伪寄存器显示应用程序中最近一次发生错误的错误代码。该寄存器与 GetLastError()函数调用完成相同的任务。对 ERR 寄存器加上,hr 修饰符后的功能将显示 32 位的出错代码，同时给出像“ Not Enough memory ”这样的出错信息。64 位 MMX 寄存器值现在显示在 Watch 和 Quick Watch 窗口中。在所有的 x86 机器上都显示这些寄存器，即使	

	这些机器本身并不支持 MMX 指令
未修饰符号	简单地说，这个特性表示你不会再在函数名中看到隐藏在某些地方的无意义的一大串字符。另外，所有的函数名都将包含可用时的参数信息
V_Table 和函数指针显示	可能时，函数指针和虚表项像文本项（符号化的）那样显示，其中用参数来代替 16 进制地址

OLE-DB 提供者模板

使用 OLE-DB 提供者模板可以开发 OLE-DB 访问接口，特别是到远程数据库的连接。该模板并不完成什么神奇的功能，它所完成的任务也就是把访问 OLE-DB 技术的过程变得容易些。OLE-DB 是微软最新的数据库技术，它用于为包含 OLE-DB 提供者的数据库管理系统（DBMS）提供高性能的数据访问。

数据绑定控件的资源编辑器改进

资源编辑器（Resource Editor）让你可以使用所有最新的 ADO 和 OLE-DB 技术来创建与数据库的连接，你需要使用新型的 ADODC 数据源控件来创建连接。另外，还需要使用新型，与 ADO 兼容的数据绑定控件显示数据。这条规则的唯一例外情况是简单的绑定控件，我们可以使用这类控件在 ADODC 和 MSRDC（Microsoft Remote Data Control，微软远程数据控件）数据源控件之间内部交换数据。

对 DocObject 包容特性的向导支持

MFC 应用向导（AppWizard）在第三步中包含了一个新的选项，该选项只在选中了 Container 或 Both Container and Server 复合文档特性选项后才有效。

利用 DocObject 包容特性（同时打开 Active Document Container 选项），你可以把应用程序支持的所有类型的文档包含在一个框架中，而不需要对每类文档分别创建不同类型的框架。如果想准确地感受一下这种技术到底如何工作，那么就请看一看 Microsoft Office Binder 吧。正常情况下，OLE 允许用户与复合文档中的一个对象进行交互，而 DocObject 包容技术让用户能够在一个框架中激活整个文档，包括菜单、工具条以及其它所支持的特性。

WizardBar 性能增强

当资源编辑器中的 WizardBar 可见时，带大量对话框和控件的（其 CLW 文件很大）工程通常慢如蜗牛。Microsoft 已经采取措施改进了这项特性的性能，从而减少了 WizardBar 对 IDE 性能的冲击。

1.3 一些缺陷

正如当今市场上所有其它产品一样，Visual C++也存在一些并不能像所期待的那样（至少像你自己所期待的那样）完成工作的缺陷。我发现的这些缺陷（某些人将它们称做“臭虫”或遗憾的变化）中没有哪个缺陷十分恐怖，以至于阻

止了你的工作，但它们依然令人讨厌。Microsoft 或许正要发行 Visual C++ 6.0 的一个或多个补丁程序，因此，我希望这里谈论的一些缺陷在你读到本文时就已经修正过了。即使 Microsoft 不能提供解决问题的替代方案，你至少也应该清醒地意识到必须修改某些代码以适应新的特性。

注释 本节并不是要罗列 Visual C++ 所有缺陷的各个方面，而是指出一些明显的问题以及避开这些问题的方法。但愿 Microsoft 也会修正一些深层次、不明显、隐藏的缺陷——即使这些缺陷在这里我并没有提到。

你首先会注意到的一件事是 InfoViewer 已经被一个非集成、基于 MSDN（Microsoft Developer Network）的帮助所取代。虽然我们依然可以加亮某个命令后得到相应的帮助，但 MSDN 的非集成特性也就意味着由外部程序提供该帮助，这也说明了现在需要更多的内存来获取帮助信息。另外，MSDN 的查找能力与已经熟悉的 InfoViewer 相比简直不可同日而语。MSDN 缺少许多 InfoViewer 提供的选项，并且基于 MSDN 的帮助还存在其它许多问题，其中最主要的问题是它要求必须有良好的 Internet 连接。总而言之，你会发现 Visual C++ 6.0 中的帮助是一大倒退——这是 Microsoft 产品升级中最失败的决策之一。

另一个问题是并非所有复制到 Bin 文件夹中的程序都会作为 Microsoft Visual C++ 6.0 文件夹中的快捷方式显示在 Start（开始）菜单中。例如，在那个文件夹中存放有 Help Workshop 快捷方式，但却没有 Hotspot Editor 快捷方式。本书中我们将多次使用这些实用程序，因此，你至少应该知道这些程序放置在什么地方，之后你就可以把它们添加到 Windows 95、Windows 98 或 Windows NT 的

Start（开始）菜单中了。再说一句，虽然 Start（开始）菜单缺少某些快捷方式并不是什么了不得的问题，但我们依然奇怪为什么 Microsoft 不把这些事说在明处。

也许部分原因是由于 2000 年日期问题，现在我们不能再使用 COleDateTime 得到两位数字的日期了。虽然该类可以接收两位数字的日期，但随后对像GetMonth 或 GetMinute 这样的成员函数的调用将导致失败。这样，对作为程序员的你，就不得不重新修改包括这些成员函数的老程序了。否则的话，尽管编写了正确无误的程序，你依然会遇到神秘的错误终止了程序的现象。

虽然界面本身并不存在自身的缺陷，但界面依然让使用多国语言的人们感到担心。Microsoft 正致力于将其所有语言工具产品的界面进行统一，也就是说除语言细节之外都是一致的，这样开发人员几乎不需要新的培训就能从一种环境迁移到另一种环境。但 Visual C++ 却是例外，它到现在还依然保持着一付老面孔。虽然对这样的界面你不需要重新学习，但对使用微软其它产品的开发者来说，他们在使用 Visual C++ 之前还必须首先熟悉该工具的集成开发环境（IDE）。幸运的是，微软计划在不久的将来改变这种状况（至少该公司中与我交流过的人们是这样说的）。

1.4 考察界面元素

在我们大量编程之前，我要花点时间讲述一下 Developer Studio 的界面，这样大家就统一口径了。你还将学习一些以前未曾用到过的新特性，例如，我们

要考察一下工具条并介绍其中的某些新特性。如果你已经是一位经验丰富的 Visual C++ 程序员，并且自认为没有什么界面元素对你来说还是新东西，那就随意跳过这一节吧。

MFC Studio 窗口元素

当使用 Visual C++ 编辑程序时，你可以把 Visual C++ 的显示界面划分成三块功能区：工具条、视图以及编辑窗口。每个功能区都可以独立于其它功能区来操作，因此，可以在各功能区之间自由地切换。图 1.1 显示了典型编辑器的显示情况以及这三个功能区的位置。

注释 本节中介绍各种窗口、视图以及工具条。随后的各节中介绍它们的详细信息。

经常用到的窗口有两种类型：文本窗口和资源窗口。

图 1.1 显示了一个典型的文本窗口。无论什么时候，当你要修改代码或编辑文本文件时，看到的就是这类窗口。

Visual C++ 通过各种手段来简化编码工作。文本窗口中首先注意到的东西是代码的颜色。例如，关键字的缺省颜色为蓝色，而注释的颜色为绿色。这种方式的彩色代码让你能够轻易地分辨出代码的性质。

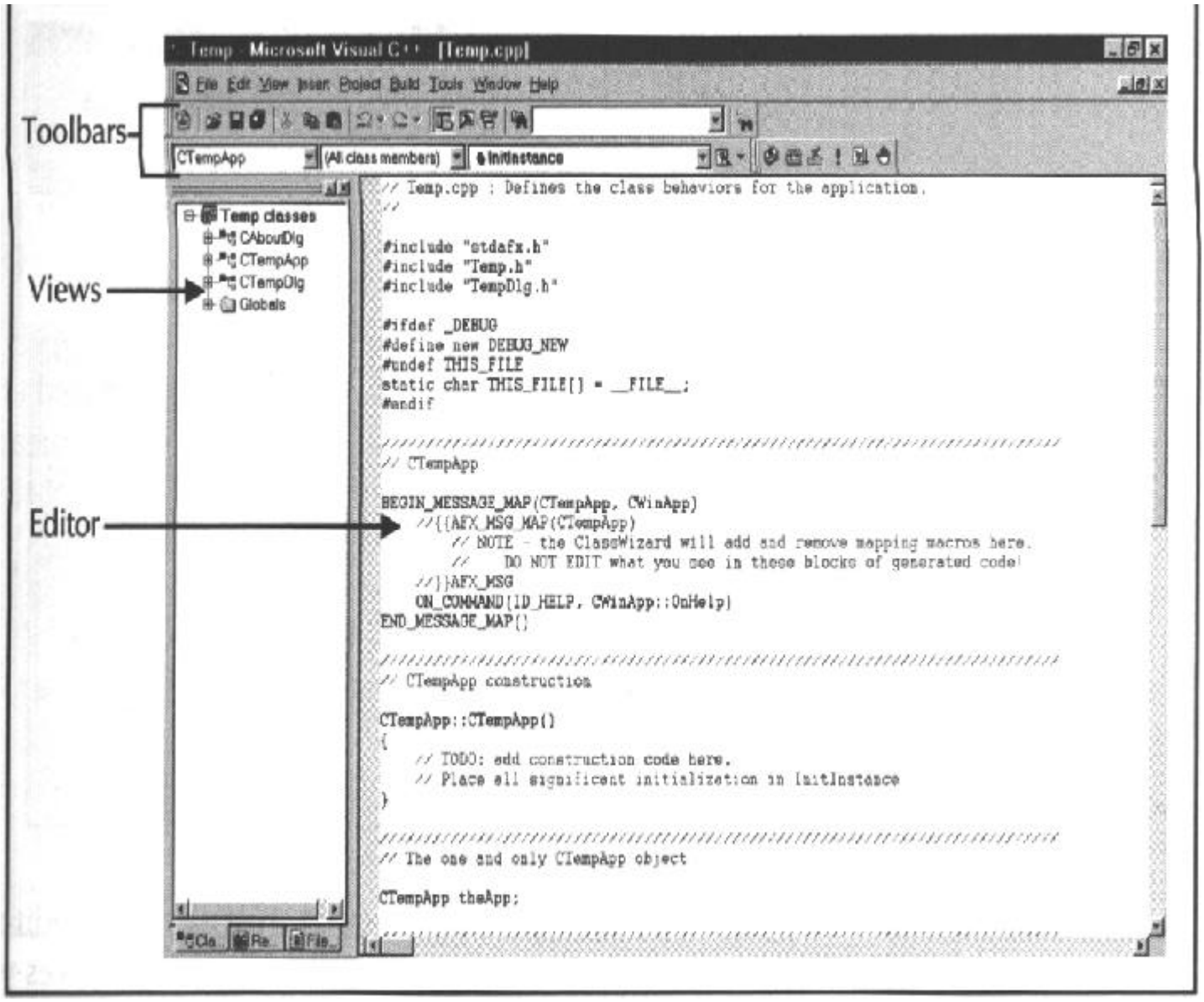


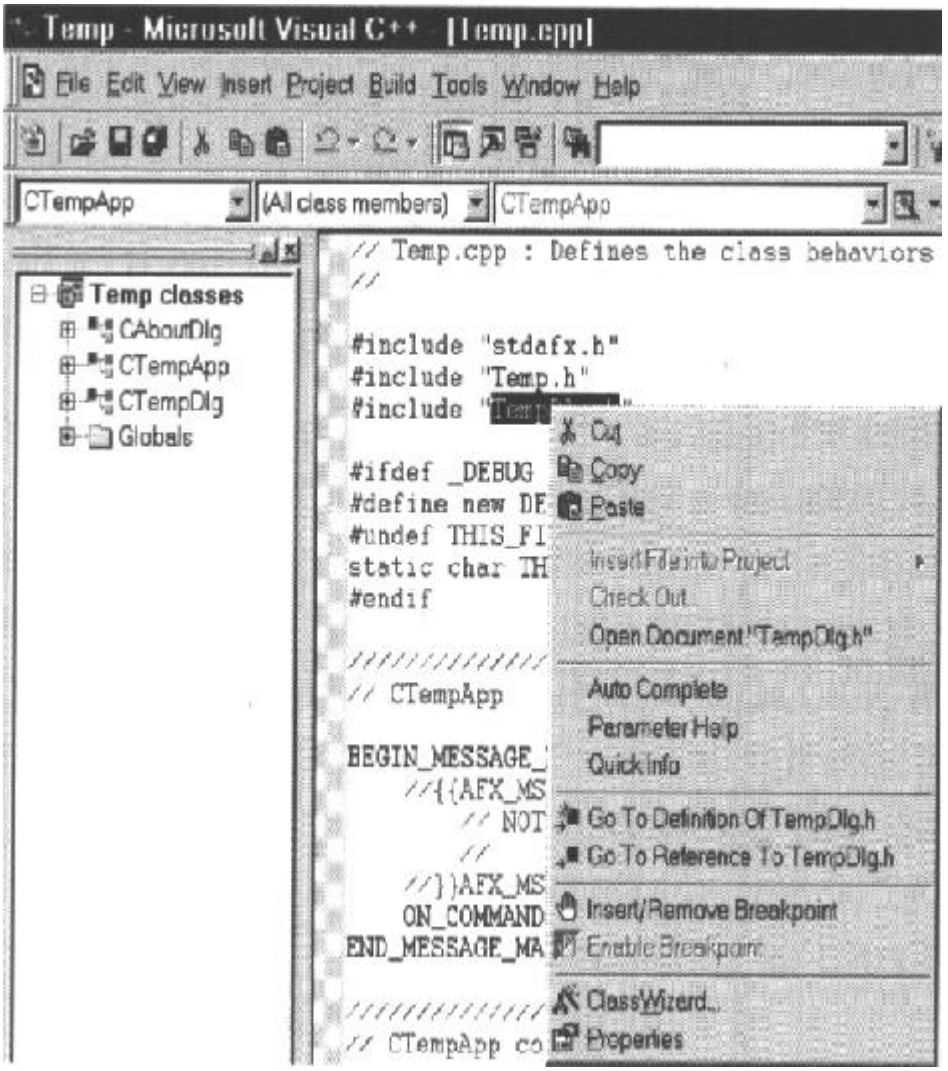
图 1.1　Developer Studio **的屏幕可以划分成三个功能区**

本窗口的左边缘是一竖条，这是 Visual C++ **放置各种符号的地方，你甚至会看到它通过颜色来区分数据和代码控制区。例如，当我们在代码中设置了断**

点后，Visual C++ 就在这个竖条上显示断点符号，当该断点可用时，断点符号为红色，否则，断点符号为白色。

文本框的各个区域中都可以通过右击来显示一个上下文相关菜单。上下文菜单中包含了可以进行基本操作的各种选项，与 Developer Studio 界面的其它地方不同，这些菜单不能直接进行修改。下图是右击文本编辑器时可能出现的上下文菜单的一个示例。

请注意，你可以像使用任何编辑器那样剪切、复制或粘贴文本。由于我已经加亮了#include 文件项，因此上下文菜单提供了一个打开该文件的机会。如果当前工程中还没有包含该文件，那么使用 Insert File into Project 菜单项就可以把该文件包含到工程中。Check Out 菜单项用于管理小组型工程，通过它可以获取对文件的控制，以便编辑文件。接下来的三个菜单项可以查找当前项更详细的信息。例如，使用 Parameter Help 菜单项可以得到函数调用的参数信息，Auto Complete 协助你完成函数调用的键入。后面的两个菜单项用于浏览工程。你可以找到某个特定项目在哪里引用、在哪里定义。由于这些项目依赖于建立工程时创建的 BSC 文件，因此，在使用这些项目前要确保最近有效地重建过工程。后面的两个菜单项——Insert/Remove Breakpoint 和 Enable Breakpoint 用于调试应用程序。最后，你可以打开 ClassWizard 来操作所选对象（本书将多次使用这种方法）或查看文档的属性。



我最后要介绍的窗口示例是你在编辑资源时看到的那种窗口，虽然资源编辑窗口比我们曾经谈过的其它窗口有更多的变化形式（无须着急，在本书的适当部分将介绍每一种资源编辑窗口），但图 1.2 **是个典型的这种窗口。**

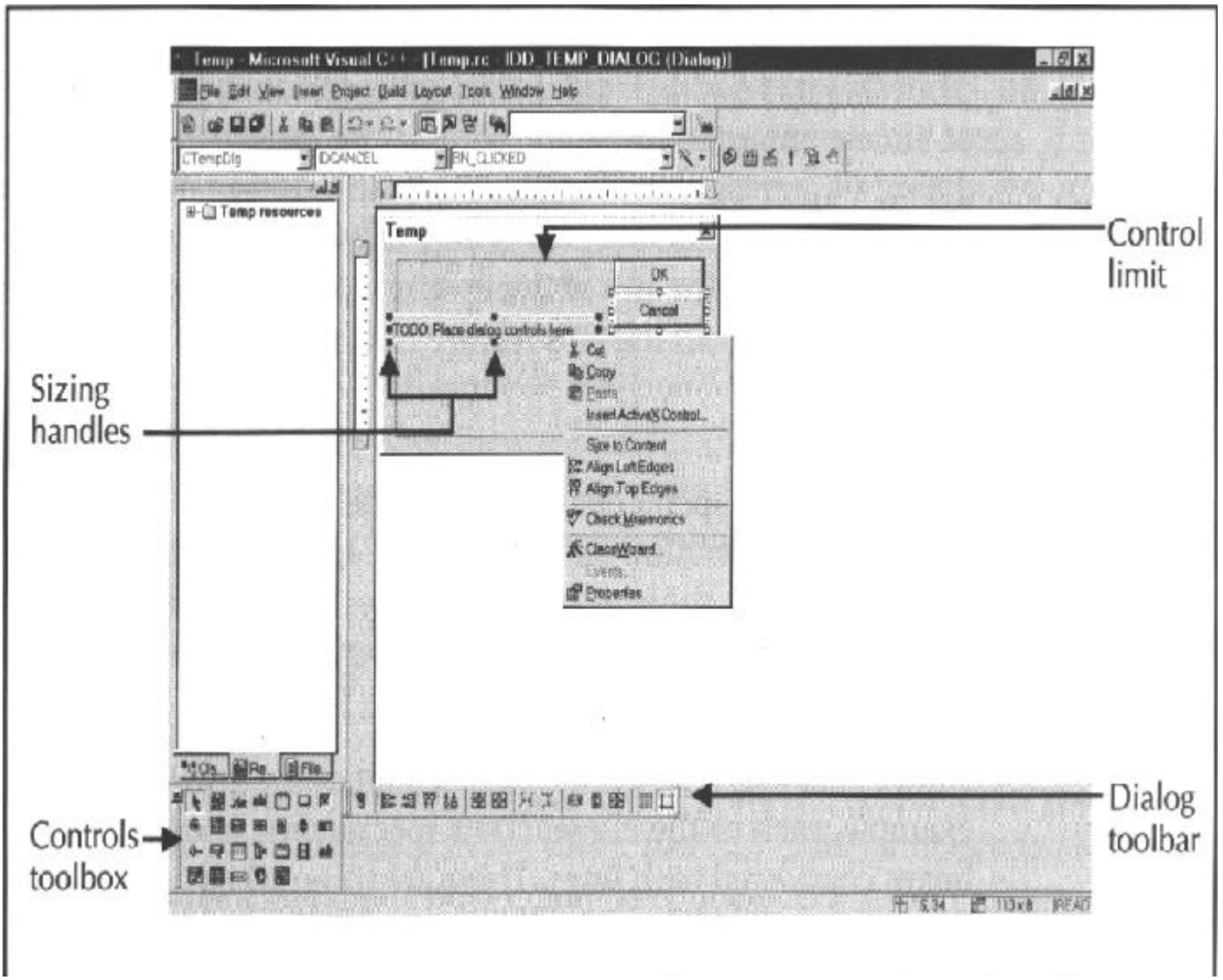


图 1.2　资源编辑窗口有许多不同的类型——这个资源编辑窗口显示了一个对话框

在上图的情况下，我们在查看对话框。这个对话框中有四个控件，其中两个控件被选中（两个选中控件的周围被改变大小的句柄所环绕）。在对话框边沿你还会看到一条暗淡的线条（缺省配置下线条的颜色为蓝色），这一线条确

定了对话框上放置控件的外边缘。

在图 1.2 的左下角，你会看到一个控件工具箱，它包含了可以放置到对话框中的各种控件，如果想在工具箱中添加新的控件，那么请使用 Project | Add to Project |Components and Controls 命令。工具箱的右边是对话框工具条，它提供了简化对话框布局排列的有用工具。例如，Make Same Size 按钮将选中的任意个数的控件与第一个选中控件等大。Test 按钮让你在对话框完成之前就可以看一看它的外观。

现在让我们讨论一下图 1.2 中显示的上下文菜单。由于前面我们已经讨论过这些菜单项的主要用途，因此这里只研究三个最重要的菜单项。使用 Insert ActiveX Control 菜单项可以把 ActiveX 控件添加到当前对话框中而无须将它实际添加到工程中。这样，你就可以在生成一大堆代码之前先看到该控件的工作效果。不过，如果你决定确实要使用该控件时，请记住以后把该控件添加到工程中。Check Mnemonics 菜单项告诉 Visual C++ 检查一下已经添加的所有控件，看是否存在同名错误或任何违反规则的情况。当做完了一个对话框后，这个检查相当重要。最后，Event 菜单项显示一个对话框，告诉我们所选对象都支持哪些事件。这个对话框还让你能够创建你想监控的任何事件的句柄。第 2 章中我们将讲述使用这一特性的方法。

类视图（ClassView）

或许类视图（ClassView）将花去你大部分的时间。该视图提供了工程中所有类的层次列表，通过扩展可以显示类中包含的细节。例如，类中将包含要编辑的成员函数等。图 1.3 显示了一个典型的类视图。

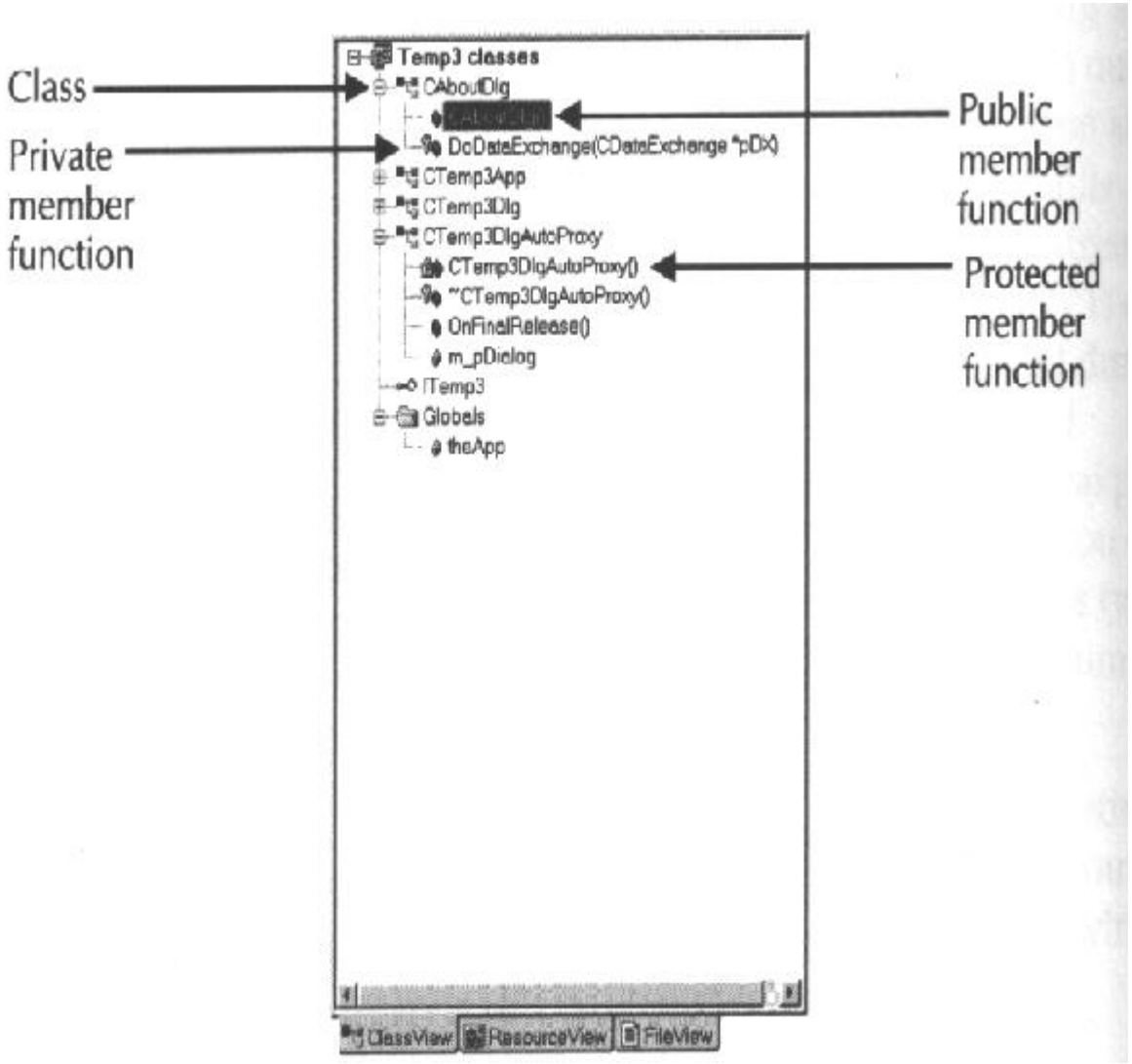


图 1.3　**在** ClassView **中显示项目中所有的类**

请注意，在层次列表的每个项目前面都有一个特殊的图标。例如，每个类的前面都有一个由线条连接起来的三个方框形图标。类视图中显示三种成员函数：第一种是公共成员函数，它的图标为一个紫色菱形框；第二种是私有成员函数，它的图标上也有个菱形框，不过菱形框的旁边还有一把钥匙；第三种是

保护成员函数，它的图标上有个菱形框和一个小锁。与此相似，变量也有三种类型，它们都使用青绿色图标。当看到一个绿色框时，你就知道是看到了某个 COM 对象的方法。类视图中还使用了其它几种类型的图标，但上面所述的六种图标是创建绝大多数工程时都需要知道的图标。

技巧 在 ClassView 窗口中右击，然后从上下文菜单中选择 Hide，可以隐藏 ClassView 窗口（或任何其它视图，只要在这些视图中按这一步骤操作）。使用 View|Workspace 命令可以重新显示 ClassView 窗口。

资源视图（ResourceView）

资源视图（ResourceView）在层次列表中列出了工程中用到的所有资源。任何图像、字符串值以及程序所需要的其它非编程部件都可以作为资源使用。图 1.4 显示了一个典型的资源视图窗口。

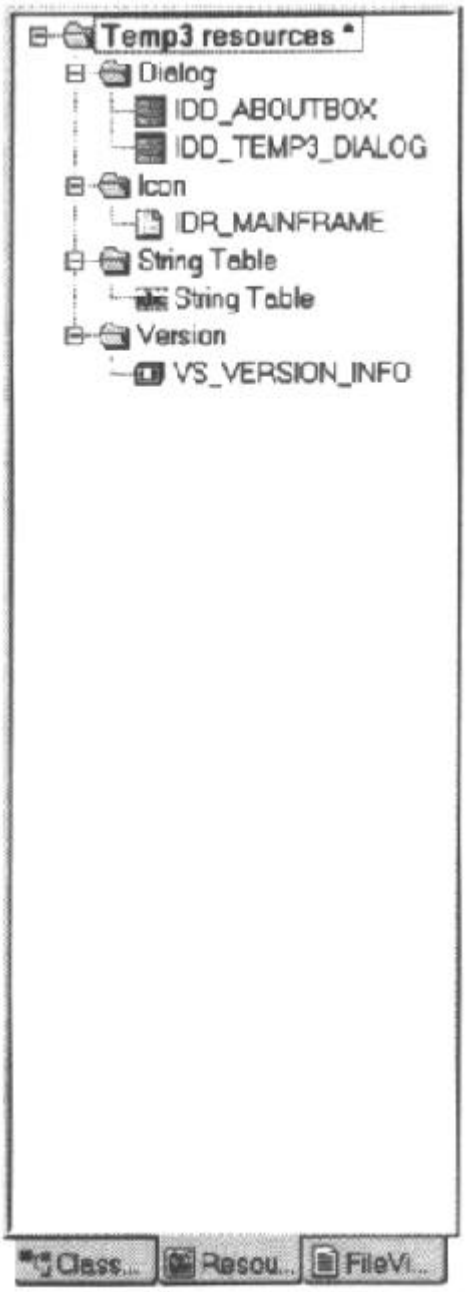


图 1.4　资源视图提供了应用程序所需资源的完整列表

Visual C++ 中可以创建的每一类资源在资源视图中都有自己的文件夹。如果你的工程中没有使用某种特定类型的资源，那么资源视图中就不会显示这种资源的文件夹。在每个文件夹中包含了工程中所用的该类资源。例如，对话框（Dialog）文件夹中包含了工程中所有的对话框，包括 About 对话框。每类资源也都使用自己的图标。

技巧 右击资源视图最顶端的文件夹，系统显示一个可以访问两类特殊对话框资源的菜单：Resource Includes 和 Resource Symbols。右击特定资源的文件夹，系统会显示添加该类新资源的上下文菜单。

文件视图（FileView）

文件视图（FileView）提供了工程中所有文件的完整列表，无论这些文件是否包含代码。图 1.5 显示了文件视图窗口的一个典型示例。请注意，列表中包含了从 ReadMe.txt 到 Visual C++ 自动生成的各种文件。

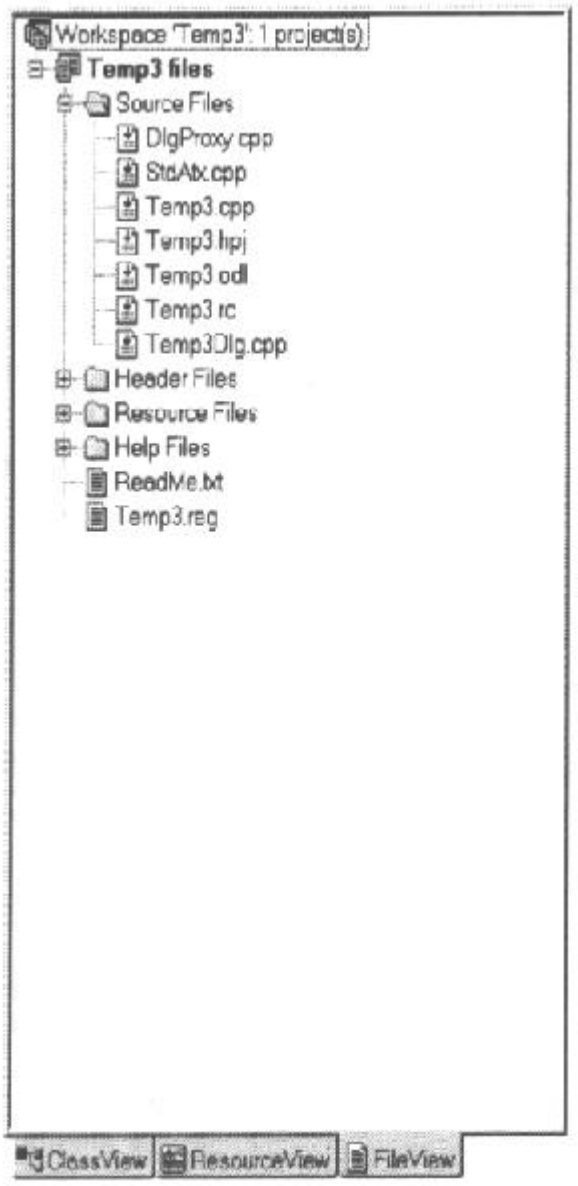
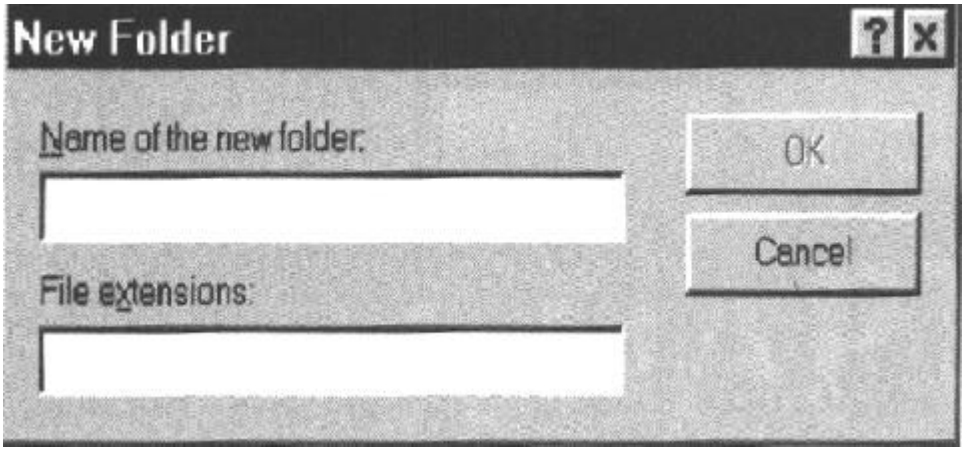


图 1.5　文件视图包含了工程中所有文件的完整列表——

即使这些文件中不包含任何代码或资源

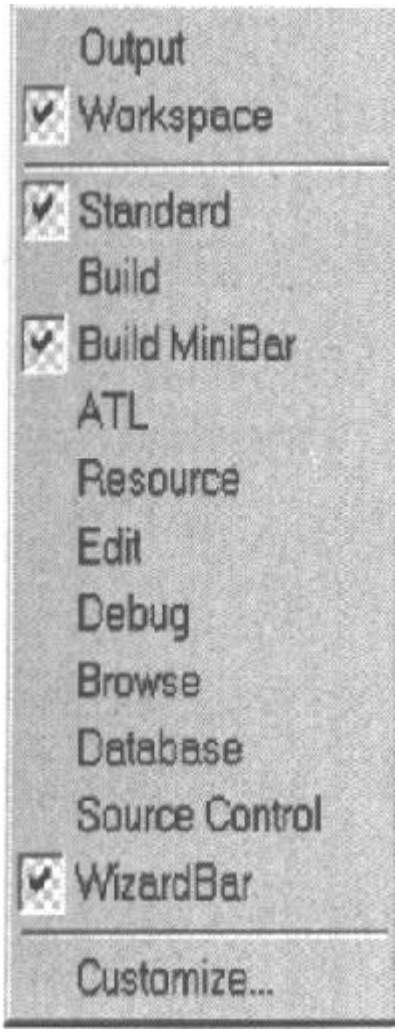
每个文件类型都有自己的文件夹，例如，所有的源代码文件都显示在 Source

Files（源文件）文件夹中。我们不仅可以把文件从一个文件夹移动到另一个文件夹中，也可以创建保存特定类型文件（根据其扩展名）的新文件夹。我通常要创建一个 Text File（文本文件）文件夹来保存所有扩展名为 TXT 的文件。创建新文件夹的方法是，右击要添加新文件夹的文件夹或工程项目，然后从上下文菜单中选择 New Folder 菜单项。此后系统将显示如下图所示的 New Folder 对话框，键入文件夹的名称以及相应的文件扩展名，然后单击 OK 按钮完成创建过程。

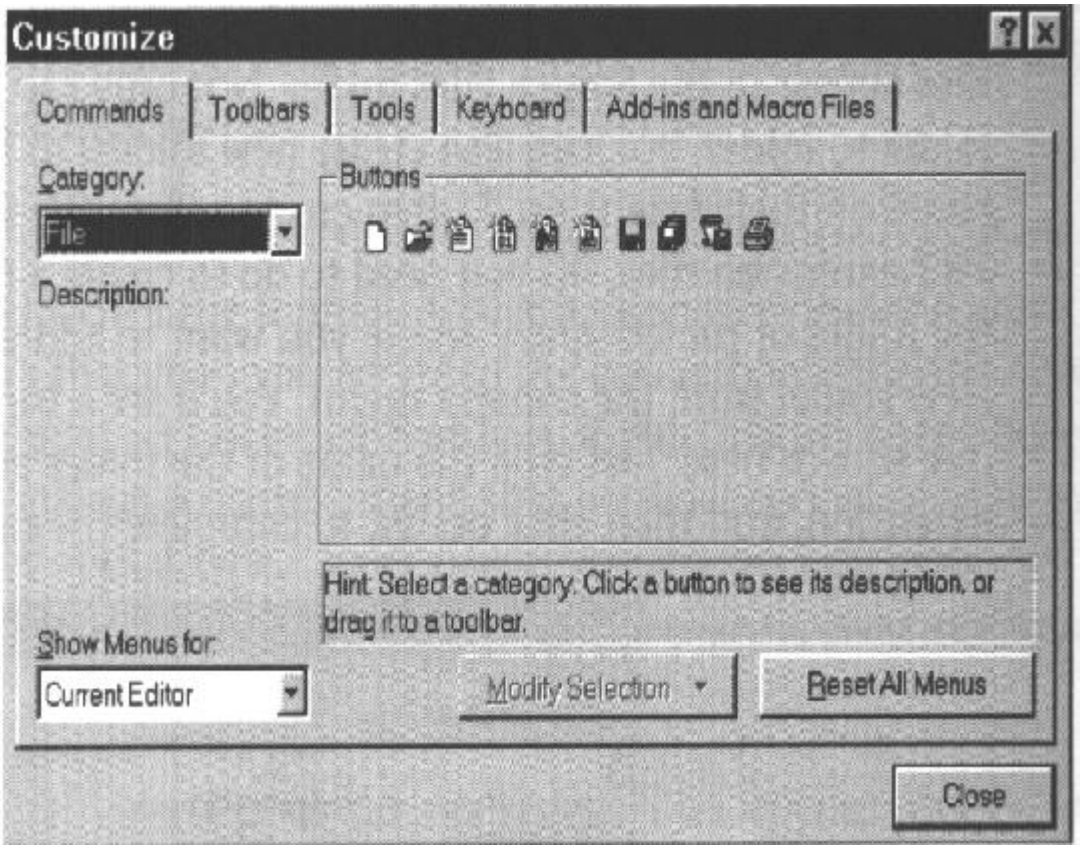


工具条

工具条（Toolbar）是我们讨论 Developer Studio 界面的最后一个话题。应该说明的是，Visual C++ 拥有比你启动 Visual C++ 后看到的工具条多得多的工具条。在工具条上的任何位置右击，你都会看到如下图所示的上下文菜单：



该菜单中列出了 Visual C++ **提供的所有标准工具条。带选中符号的菜单项对应的工具条当前显示在工具条区域中。如果想把某个菜单项对应的工具条添加到工具条区域，只需要单击该菜单项即可；反过来，如果想在工具条区域中去掉某个工具条，也只需要通过在上下文菜单中单击对应的菜单项、去掉其选中符号即可。**



上下文菜单中还有一个 Customize **菜单项值得注意。选择该菜单项后你会看到如下图所示的对话框：**

Customize **对话框包含了几个让你重新配置** Developer Studio **界面上绝大多数部件的选项卡。然而，这里我们最感兴趣的是** Commands **选项卡。对每个菜单和工具条它都包含了一系列的图标（代表对应的命令）或文本命令。如果想给现存工具条上增加一个命令，只需要抓住其图标（或文本命令）并把它拖曳到工具条上，这样，所选命令就显示在工具条上，你可以更快速地访问它们了。如果选择了某个现在还没有相应图标的命令，那么系统会显示** Button Appearance **对话框，在这个对话框中可以为命令选择一个图标。**

同样的功能对菜单项也一样起作用。你需要做的工作就是打开要修改的下拉菜单，在 Commands 选项卡中抓住要添加到菜单中的命令，然后将其拖曳到菜单项中希望的位置。在这种情况下，你看到的总是命令的文本版本而不是图标。

从工具条或菜单中删除不需要命令的方法也很简单。简单地抓住不需要的命令，并把它拖曳到 Customize 对话框中。该命令将在菜单或工具条中消失，当然，你随时都可以重新把它添加到菜单或工具条中。

你也可以根据需要移动工具条的位置。单击工具条左边缘的双线竖杠，然后把工具条拖曳到所需位置。与此相似，如果你不喜欢某个菜单的当前位置，那么抓住它（在 Customize 对话框打开的情况下）并把它移动到所需位置。

第2章　建立基本应用程序

学习一门新语言，甚至更新你曾经用过一段时间的某门语言的知识，从来就不会像郊游野餐那么容易。旧的语言已经习惯，而新的语言总是包含一些不熟悉的东西，用起来别别扭扭。然而，如果你想紧跟当今商业化世界的潮流，就必须学习新的语言（或最低程度上更新老语言的知识）。

那么，如何以最短的时间跨越学习曲线底部呢？我认识的许多程序员都有一个用于学习新语言的基本应用程序。实际上，真正高水平的程序员都有几个基本的应用程序，称之为“测试套件”，其实际用途就是学习和测试新的语言。这类应用程序的功能相对比较全面，也可以再增加其它的功能，否则的话，你就不知道是否真正学会了这门语言。

上一章已经考察了Visual C++的功能，本章打算给读者一个Visual C++编程的初步印象。即使你是个C++高手，也同样值得花点时间“玩一玩”Visual C++的这个新版本，这样才能确定如何使用该产品。建立这类“测试套件”将帮助你了解如何使用Visual C++以及Visual C++能为你做些什么。

当然，我们要花点时间建立能够使用的应用程序。为简单起见，对绝大多数情况来说，我把应用程序完整功能实现的任务留给了读者。然而，不要忘记本章的真正目的是演示——当样本程序与实际工作十分相似时，或许就到了抛

弃样本程序、开始实际工作的时候了。

Web 链接 任何严肃的程序员都会花点时间在 Internet 上学习新的编程技巧。Microsoft 主持了各种 Visual C++的新闻组，其中一些相当专业。最通用的新闻组是 microsoft.public.vc.language。如果你想了解 ActiveX 技术的进展情况，或许应该到 microsoft.public.vc.activextemplatelib 看一看。查找数据库信息的一个好位置是 microsoft.public.vc.database。最活跃的新闻组之一是 microsoft.public.vc.mfc，这里专门讨论 Microsoft Foundation Classes（MFC，微软基础类库），当然，还有另外两个与 MFC 相关的新闻组：microsoft.public.vc.mfc.docview 和 microsoft.public.vc.mfc.macintosh。最后，不要忘记看一看 microsoft.public.win32.programmer 下的一般 Windows 编程新闻组（它下面有许多与程序员有关的新闻组，因此，你需要选择最适合自己需要的新闻组）。

2.1 了解应用程序类型

Visual C++有能力创建你能够想像到（以及有些你想不到）的任何应用程序。然而，从整体上看，应用程序可以分为五种类型，这就是本文首先要集中讨论的内容：

- 控制台应用程序适用于你真正需要与遗留系统保持某种兼容性或不需要为用户提供全功能操作界面的情况。
- 基于对话框的应用程序通常是实用程序的专利，也适用于极小型不需要菜单系统的应用程序。
- 单文档应用程序适用于操作自有数据的简单应用程序，比如记事本或小型数据库前端应用。这类应用程序也需要某种类型的菜单系统。
- 多文档应用程序是提供完整功能的应用程序，比如字处理程序或电子表格。由于多文档应用程序是 C++编程中十分复杂的部分，因此，当你考虑建立这类应用程序时，应该在 Visual C++的灵活性与诸如 Visual Basic 之类的快速应用开发工具提供的速度之间进行权衡。
- 基于 HTML 的应用程序是 Visual C++ 6.0 新增的应用程序类型。它们操作某种类型的数据（像单文档或多文档应用程序那样），但与 Internet 结合在了一起。作为标准编辑器的替代物，你的用户会看到 Web 浏览器风格的前端应用。

注释 请注意本章我们在讨论应用程序。Visual C++有能力创建各种不同类型的代码。使用 Visual C++不仅可以创建 DLL、ActiveX 控件、ISAPI 扩展程序、设备驱动程序、像屏幕保护器之类的后台应用程序，甚至也可以扩展 Visual C++本身。本章中我们只讨论一般的应用程序，随着本书的展开，我们也会介绍许多其它类型的代码。

控制台应用程序

我前面已经介绍过控制台应用程序的概况，但由于这一介绍过于粗略，以至于并未真正告诉读者控制台应用程序到底是什么样的。控制台应用程序具备 DOS 风格的窗口外观，而不是读者更熟悉一些的 Windows 风格窗口。控制台应用程序使用等距字体，就像你在 DOS 窗口中看到的那样，在控制台应用程序中可以使用标准 C 函数完成输入输出，比如 printf 和 scanf()。然而，从内部上讲，控制台应用程序确实是个 Windows 应用程序。下图是个控制台应用程序的典型示例（本章后文中我们将实际创建一个这样的程序）。

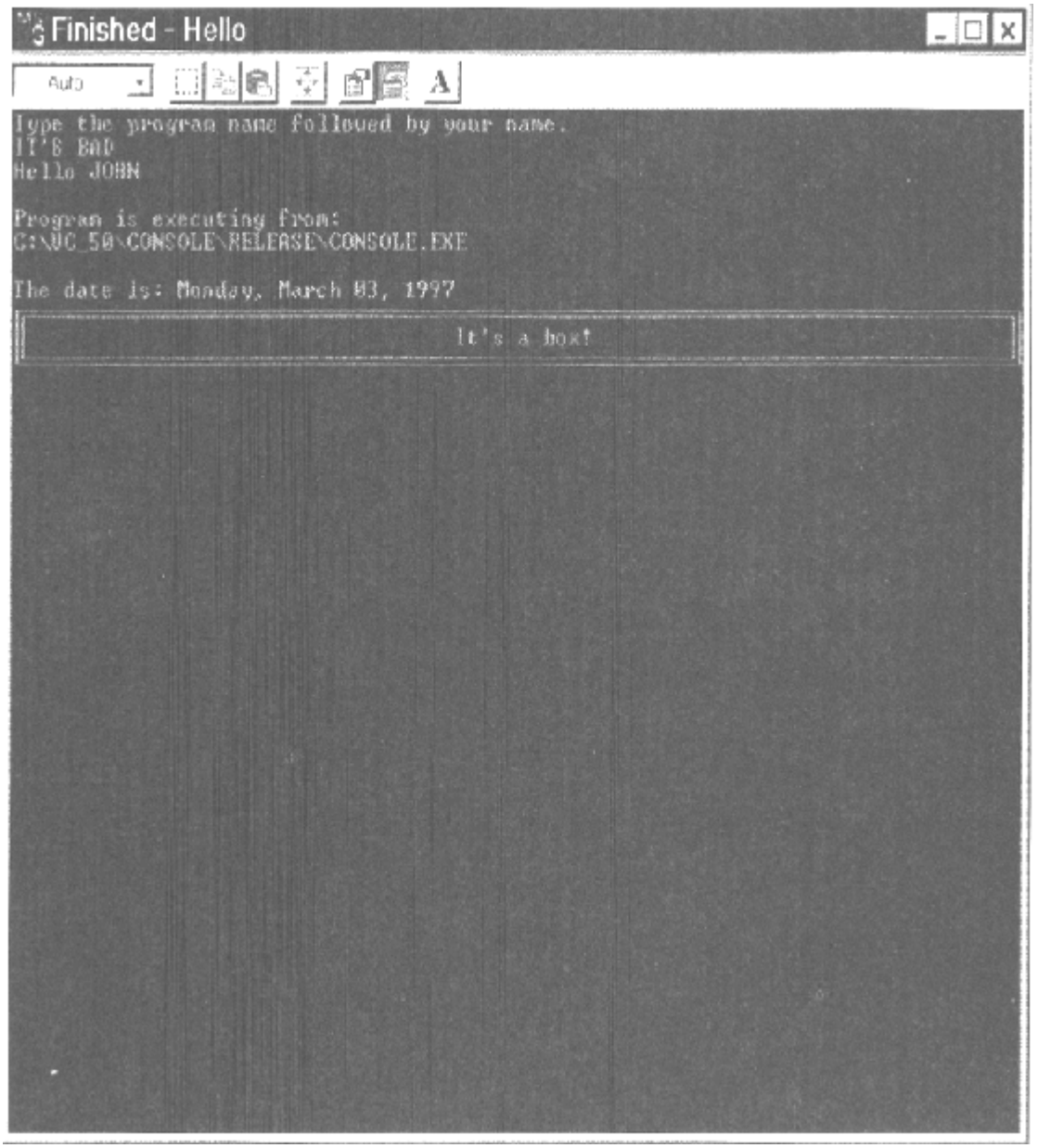
注　使用控制台应用程序可以把某些功能型 DOS 代码转移到 Windows 系统之下。

那么，为什么要创建这种杂交怪物呢？假设说你有一些工作得很好的老的遗留代码，但它们是 DOS 下的应用程序。你想把这个程序迁移到 Windows 系统下，但又没有时间把所有代码都转换成 Windows 调用，那么控制台应用程序提供了一个折中的解决办法。在这类应用程序中你可以使用老的代码（至少是其中的一部分），但在 Windows 环境中执行这一程序。

注　从长远的观点看，创建控制台应用程序并不一定会节省时间和精力——没有谁能保证让你使用所有现存代码。

使用控制台应用程序还有另外一个好处。当把应用程序从 DOS 搬到 Windows 中时，外观上就不一样。没有什么办法填平两者之间的鸿沟，从用户的观点来看，你引入了一个完全不同的应用程序，需要做出计划重新培训每一个工作人员及一切相关的事情。另一方面，控制台应用程序外观上与 DOS 应用程序十分

相似，每个人都已经相当熟悉，因此或许你并不需要花多少精力进行配置。你会感到烦恼的唯一的一件事就是如何让每个人都习惯以不同方式运行应用程序。



然而，不要以为控制台应用程序是每个 DOS 应用程序迁移到 Windows 环境

的理想途径。本质上两种方式下你都要创建新的应用程序，其间的唯一差别是可重用代码数量，因此也表现在创建新的应用程序所耗费的时间。事实上，如果你决定采用控制台应用程序这种方法时，只要考虑一下就会发现，你终将编写两个应用程序。绝大多数公司发现他们终将承受把 DOS 应用程序转换到Windows 应用程序所付出的代价。

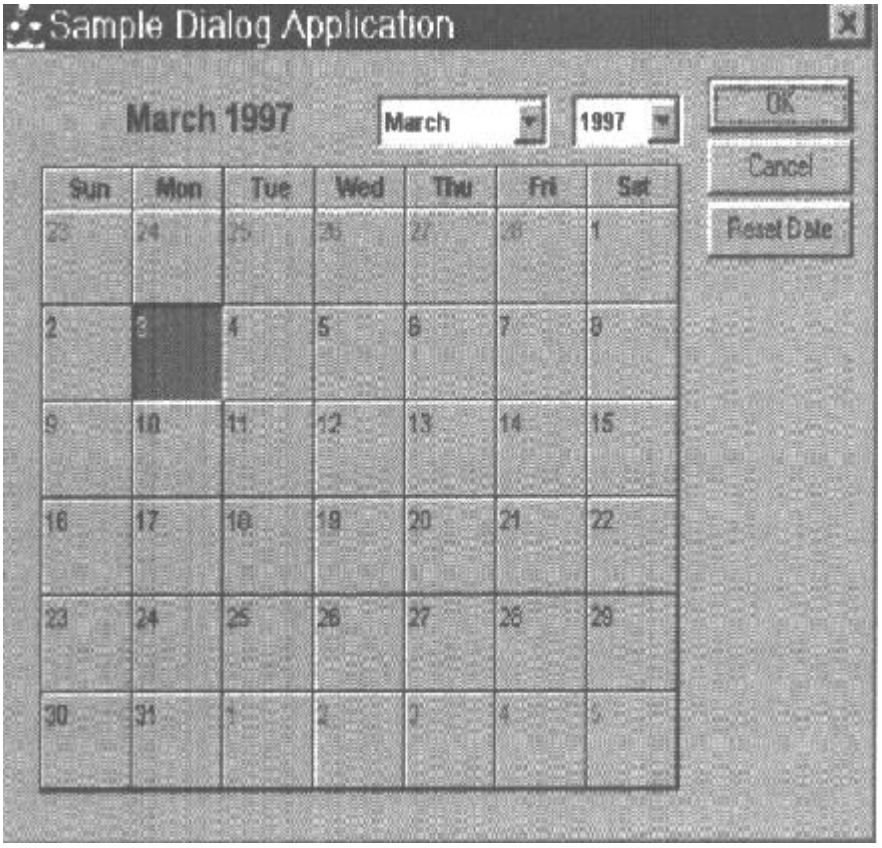
当然，如果 DOS 应用程序在 DOS 窗口中运行得相当好时，你也必须斟酌一下重新编写控制台应用程序是否明智。无疑地，控制台应用程序可以使用某些 Windows 服务，但或许所需的一切服务已经在 DOS 应用程序中实现了（你需要访问某些 MFC 函数以增强应用程序的功能并改进应用程序的易用性）。向你的 DOS 应用程序中添加越多的 Windows 功能，控制台应用程序的可采用性也变得越低。

可移植性 可以安全地假定使用控制台应用程序能够把 DOS 应用程序的“商务逻辑”迁移到 Windows 环境，也可以把某些显示和打印功能进行迁移。然而，想通过控制台应用程序使用一组代码同时实现 DOS 和Windows 下应用程序的假定从来都不是安全的——这两个环境的差别实在太大了，当把应用程序从 DOS 迁移到 Windows 时总需要进行某些修改。

基于对话框的应用程序

许多人把配置屏幕、About 对话框之类的应用与基于对话框的应用程序联系

起来，而把其它功能要求归类于全功能应用程序。基于对话框的应用程序在编程世界中也占有一席之地。它们对实用程序类应用程序极为有用，在这类应用程序中通常只需要显示少量的数据和获取少量的用户输入。下图是本章后面要创建的一个基于对话框的应用程序的示例：



技巧 当确定是创建基于对话框的应用程序还是创建基于窗口的应用程序时，需要考虑实用程序。如果你的应用程序适合作实用程序，那么基于对话框的应用程序界面或许是良好的选择。另一方面，如果打算在应用程序中添加众多的特性或者需要用户进行大量的交互操作，那么应该考虑选用基于窗口的界面。在做出决定时一定要考虑未来对应用程序的扩充问题——今天做出的错误选择将在明天的重新实现中付出沉重的代价。

那么，到底是什么原因使基于对话框的应用程序比基于窗口的全功能应用程序更好呢？最重要的原因之一是程序规模。你可以创建同一个应用程序的两种版本，一种使用对话框界面，另一种使用窗口界面。每次做这样的实验时你都会发现对话框版本的程序更小一些。除了节省资源外，对话框版本的应用程序的加载速度也更快些。基于对话框的应用程序比完成相同功能的基于窗口的应用程序更简单有效。

你还会发现创建基于对话框的应用程序的速度也很快。基于对话框应用程序的特点就是规模小、效率高。当你发现需要在这类应用程序中增加大量功能和特性时，或许你一开始就选错了要创建的应用程序的类型。基于对话框的应用程序通常避免使用菜单和其它基于窗口应用程序为提供友好界面所需的部件。更少的特性减少了程序员的开发和调试时间。显然，任何有利于提高程序员效率的方法都值得一试。

注 把过多的控件塞进基于对话框的应用程序的做法只能使该应用程序既笨拙又难以使用。

基于对话框的应用程序并不一定要承受缺乏重要特性的遗憾。例如，你可以把基于对话框的应用程序创建成完美的 OLE 服务器。在这方面 Visual C++向导会为你提供相应的帮助，因此，在应用程序中添加 OLE 支持只需要你多做一丁点工作。

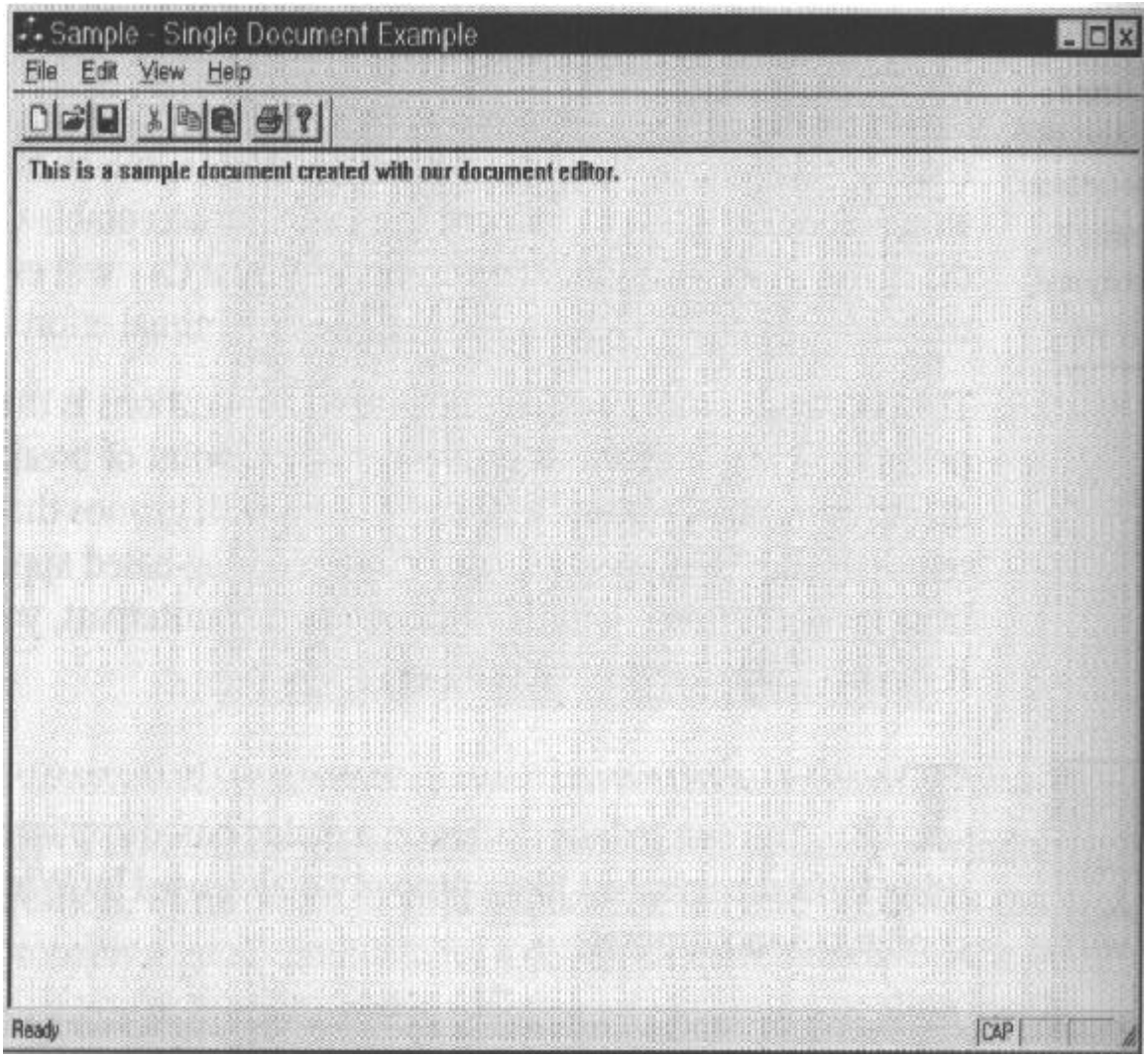
使用对话框应用程序唯一存在的问题是某些程序员感到难以划分对话框。我曾经见过一个基于对话框的应用程序，其上充满了让人不知所措的各种小玩意。基于对话框的应用程序看上去会比同等功能的基于窗口的应用程序拥挤些，但也不应该把它们拥挤到谁也不能使用的地步。

技巧 就像 Windows 中用于配置的属性对话框那样，你可以使用选项卡来降低基于对话框应用程序界面的拥挤程度。

单文档界面应用程序

单文档界面应用程序是像 NotePad（记事本）或 Microsoft Paint（画笔）这样的应用程序。它一次处理一个文档，降低了编程的复杂度并减少了运行程序时所需的资源。对某些小型应用（比如文本编辑器或小型图像编辑器）可以采用这种类型的窗口应用程序。单文档界面应用程序允许用户与其创建的文档进行全面的交互，但通常不如多文档界面的应用程序耐用。另外，单文档界面应用程序至少要比多文档界面的应用程序少一个菜单——就是用于选择要编辑文档的 Windows 菜单。

注 对只需要用户进行少量交互的小型应用程序来说，可以采用单文档、基于窗口的界面。



与基于对话框的应用程序相似，单文档应用程序也可以创建成 OLE 服务器。实际上，这类应用程序也可以作为 OLE 客户程序，尽管极少有程序员把这种能力添加到他的应用程序中。下图是本章后面要介绍的一个单文档应用程序示例。请注意这个示例可以作为 OLE 的客户端。

注 通过把应用程序的基视图类选择为 CHtmlView，可以把单文档界面的应用程序转换成简单的 Web 浏览器。

不幸的是，单文档界面的应用程序与基于对话框的应用程序有相同的问题——用起来太复杂了。我还记得以前使用老版本 CorelDRAW 时的问题。每当我想查看一幅图案时，不得不在查看之前首先关闭当前打开的文档。这种限制使得 CorelDRAW 比它应该提供的方法要难用一些。例如，我在比较两幅图案时浪费了太多的时间（所幸的是，Corel Systems 在当前版本的 CorelDRAW 中已经纠正了这一缺陷）。

技巧 当操作数据库管理系统时，单文档、基于窗口的应用程序工作的相当完美，其原因相当简单，极少有用户需要同时打开多个数据库。即使他们需要同时打开多个数据库，数据库本身的使用规则也减少了用户本身访问多个数据库的可能性。正常情况下，你需要以可编程方式控制对各种数据库元素的访问，并把结果显示给用户。

多文档界面应用程序

现在我们来谈谈多文档界面应用程序。使用这种类型基于窗口的应用程序可以创建像字处理程序或电子表格那样的应用程序。例如，Microsoft Word 和 Microsoft Excel 都是多文档应用程序的示例。如果你想一想，就会发现，文本编辑器具备有限吸引力的原因正是由于其一次只能打开一个文档。人们需要在文档之间进行比较，这就是多文档界面的应用程序不仅幽雅而且在众多情形下需要的原因。

注 通过把应用程序的基视图类选择为 CHtmlView，可以把多文档界面的应

用程序转换成简单的 Web 浏览器。

多文档界面的应用程序通常也都具有多种功能（你可能会走向另一个极端，看一看人们对当今厂商生产的主流产品臃肿特性的抱怨也就知道了）。文本编辑器可以提供十分简单的查找功能但并不提供替换文本的任何方法。而全功能的字处理程序则把查找和替换作为标准功能来对待。

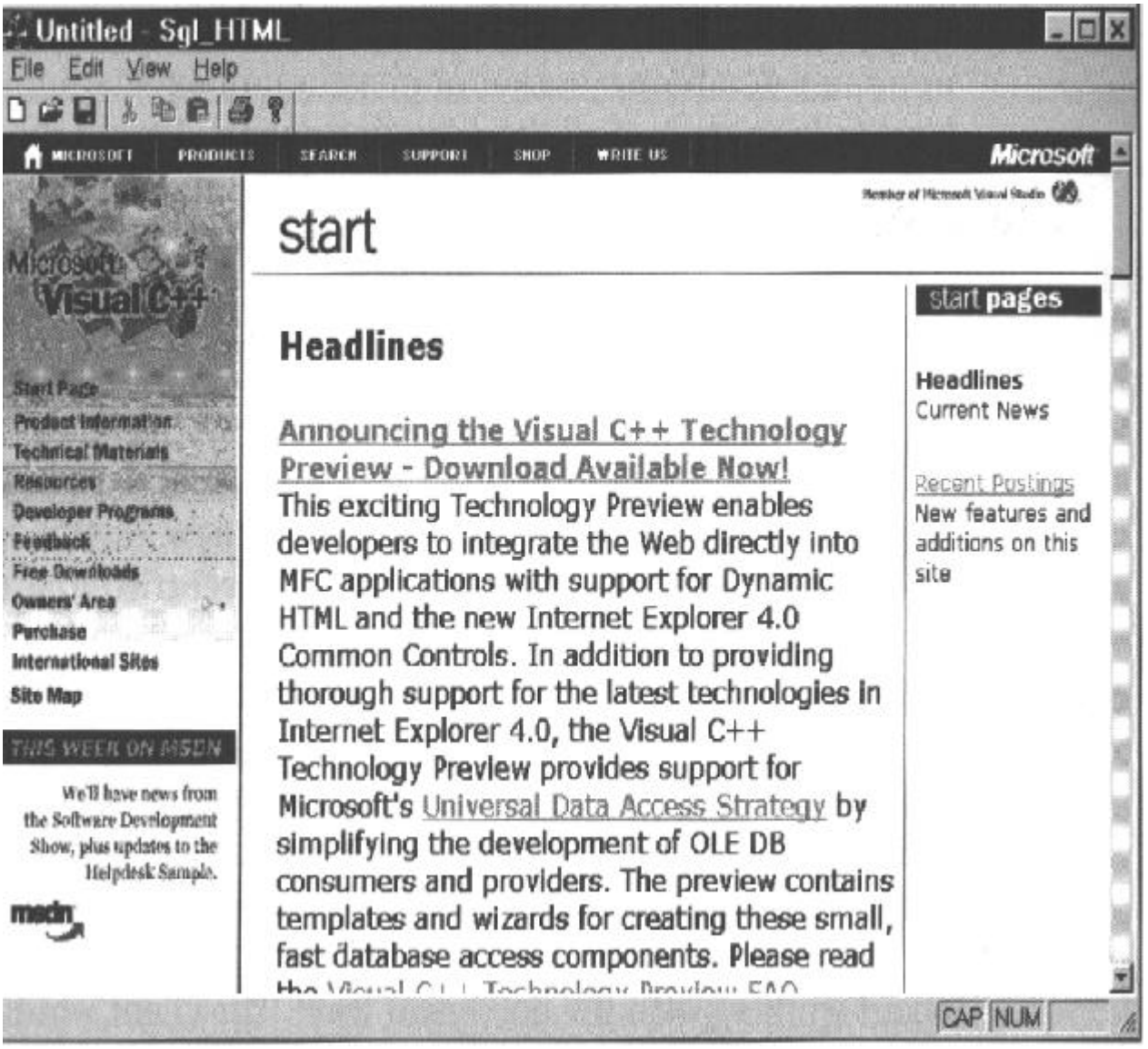
多文档界面应用程序的缺点就在于它处理多个文档。同时处理多个文档的能力也意味着需要更多的额外编程工作。你并不只是要跟踪所有打开的文档，也要提供 Windows 菜单来管理特殊的程序特性，比如要考虑屏幕划分问题。你还需要决定是否允许用户同时显示多个文档。像最小化其中一个文档，而最大化另一个文档这样的任务也需要额外的程序代码。总而言之，即使在开始编写多文档应用程序之前，就需要做大量的编程准备工作了。

当然，多文档界面的应用程序也有不少的缺点。例如，如果你以前把 Word 用做过 OLE 服务器，你就会知道，在另一个应用程序中单击链接之后，每次打开这个庞大的应用程序时都要等待很长时间这种烦恼了。而且，你或许也经历过内存不够的困惑。在最近之前，每当你要使用 OLE 时，你就必须有足够的内存同时运行两个应用程序（客户和服务器）。所幸的是，通过让服务器接管客户端窗口的方法，Microsoft 已经降低了这类内存需求，现在只负责操作文档本身了。客户窗口为服务器菜单和工具条提供了框架，因此不会再浪费额外的内存空间了。

基于 HTML 文档的应用程序

Visual C++ 6.0 提供了一种新的应用程序类型，但你在 New 对话框的 Project 选项卡中却找不到它的踪影。你可以通过 MFC AppWizard 第 6 步的对话框创建基于 HTML 文档的应用程序，本章中我们将花费几节的篇幅讨论这方面的内容。上述对话框 Base Class（基类）组合框中包含了一个 CHtmlView 选项，正是使用这个选项来创建这种新型的应用程序。

那么，基于 HTML 文档的应用程序有什么优点呢？想一想创建自己定制的 Web 浏览器的好处吧！你可以把这个浏览器设置成自动浏览公司 Web 站点并限制用户访问 Web 上非商务站点的方式。由于定制的浏览器不需要具备全功能浏览器所有的通用功能，因此，定制浏览器会减少内存需求和磁盘空间需求。换句话说，你可以建立一个提供浏览器所有功能、而又不存在它们所含问题的程序环境（至少在访问公司 Web 站点上是这样）。下图是个用于访问 Web 站点的基于 HTML 文档的应用程序的示例（显示的是 CHtmlView 类提供的缺省 Web 站点）。



不管怎么说，这种新型应用程序比你原先想像得更有价值。例如，你可以把 CHtmlView **类添加到现存应用程序中，让它能够访问基于** Web **服务器的帮助桌面（在第** 15 **章的“给你的应用程序添加基于** HTML **的帮助”一节我将介绍基于** HTML **的帮助）。作为创建标准帮助文件并把它添加到应用程序的一种替代方法，你可以创建十分专业化的** Web **浏览器，并把它置入到应用程序中。**

基于 HTML 的帮助的优点十分明显。使用老的帮助文件就意味着一旦把应用程序交付给用户或在整个公司内分发后，你就不能够轻易地更新帮助文件。而更新 HTML 帮助则简单到只需要在 Web 服务器上修改文件即可。另外，使用 Microsoft Help Workshop 还需要做一些额外的工作。而基于 HTML 的帮助则既不需要编译器，也不需要特殊工具，只要有个文本编辑器即可（理论上说，在编写巨型帮助文件时，你需要个专门为操作 HTML 而设计的编辑器）。

HTML 帮助也有一些缺陷。一方面，难以在基于 HTML 的帮助中建立能够满足需要的查询功能。由于查询用户所需的信息与建立这些信息处于同等重要的位置，因此，基于 HTML 的帮助并不适合于初级用户。另外，基于 HTML 的帮助必须建立 Internet（至少为内部网）链接。如果你的公司中有许多旅途中的用户，那么试图建立 Internet 链接或许并不现实。当然，总可以复制所需的 HTML 文件，但这与以前的帮助形式存在相同的问题：过期的帮助文件。

这种新的应用程序类型还有许多其它的用途——多的这里无法罗列。需要记住的重要一点是，从 Internet 访问中能够得到好处的应用程序从 CHtmlView 类的使用中也会得到好处。利用这个类，可以完成从销售代表处远程更新公司数据库到让用户易于登记产品之类的一切任务。换句话说，CHtmlView 类为你和你的用户打开了一个新的世界。

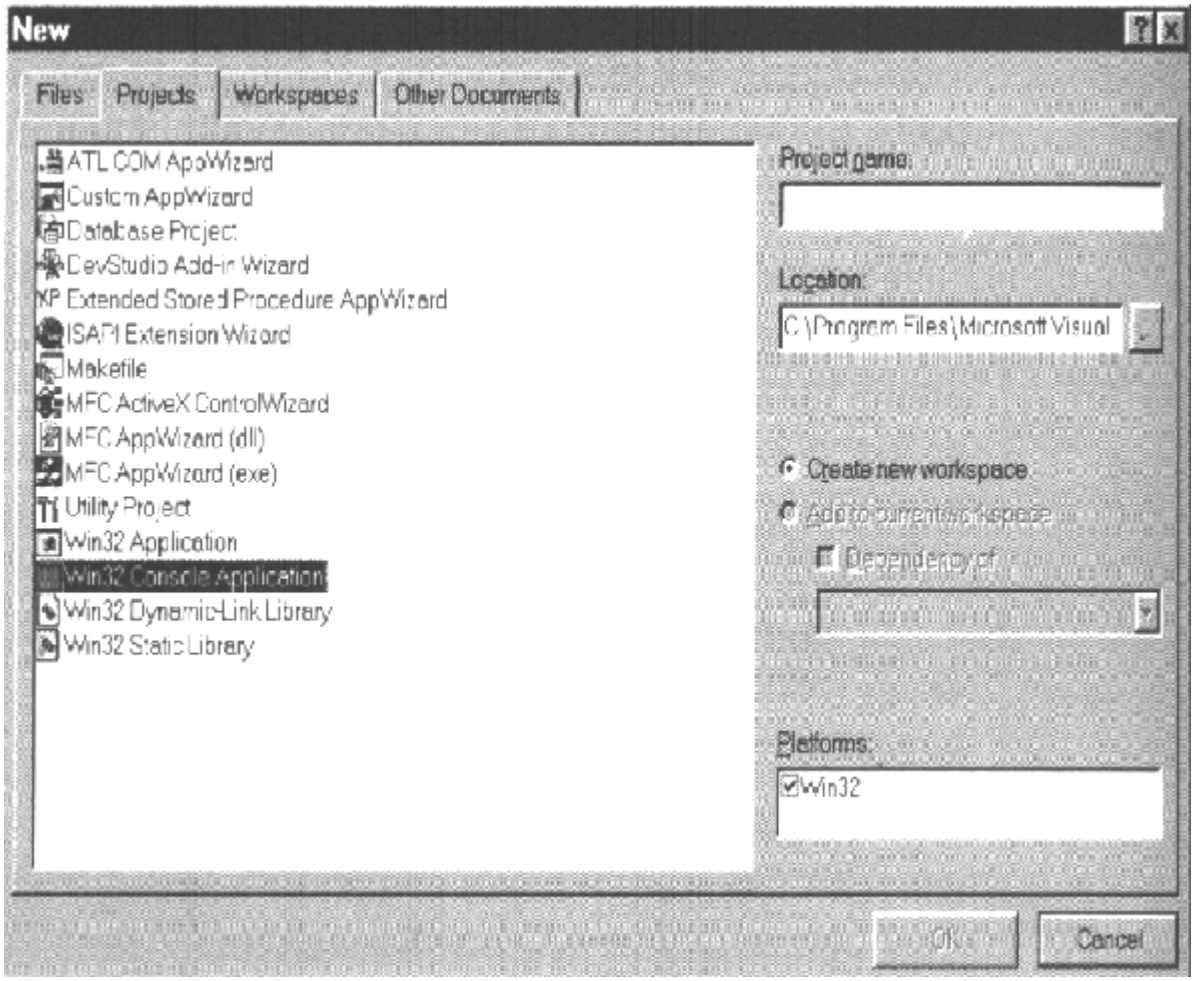
2.2 编写控制台应用程序

如前所述，控制台应用程序让你能够把应用程序的商务逻辑从 DOS 环

境迁移到 Windows 环境。你也可以迁移某些（甚至是绝大多数）显示逻辑，但也需要一些 MFC 提供的功能。本质上讲，控制台应用程序就像添加了一些特性的 DOS 应用程序。编码之后，需要彻底地测试控制台应用程序，以确保从 DOS 迁移到 Windows 的各种特性依然能够正确地工作。

让我们探讨一个相当简单的控制台应用程序，看看在这个程序中你能做些什么。这个示例中，我们并不把程序的功能看得很重要，只是要了解应该如何完成示例。当然，第一步的工作是创建程序框架。按下述步骤进行操作：

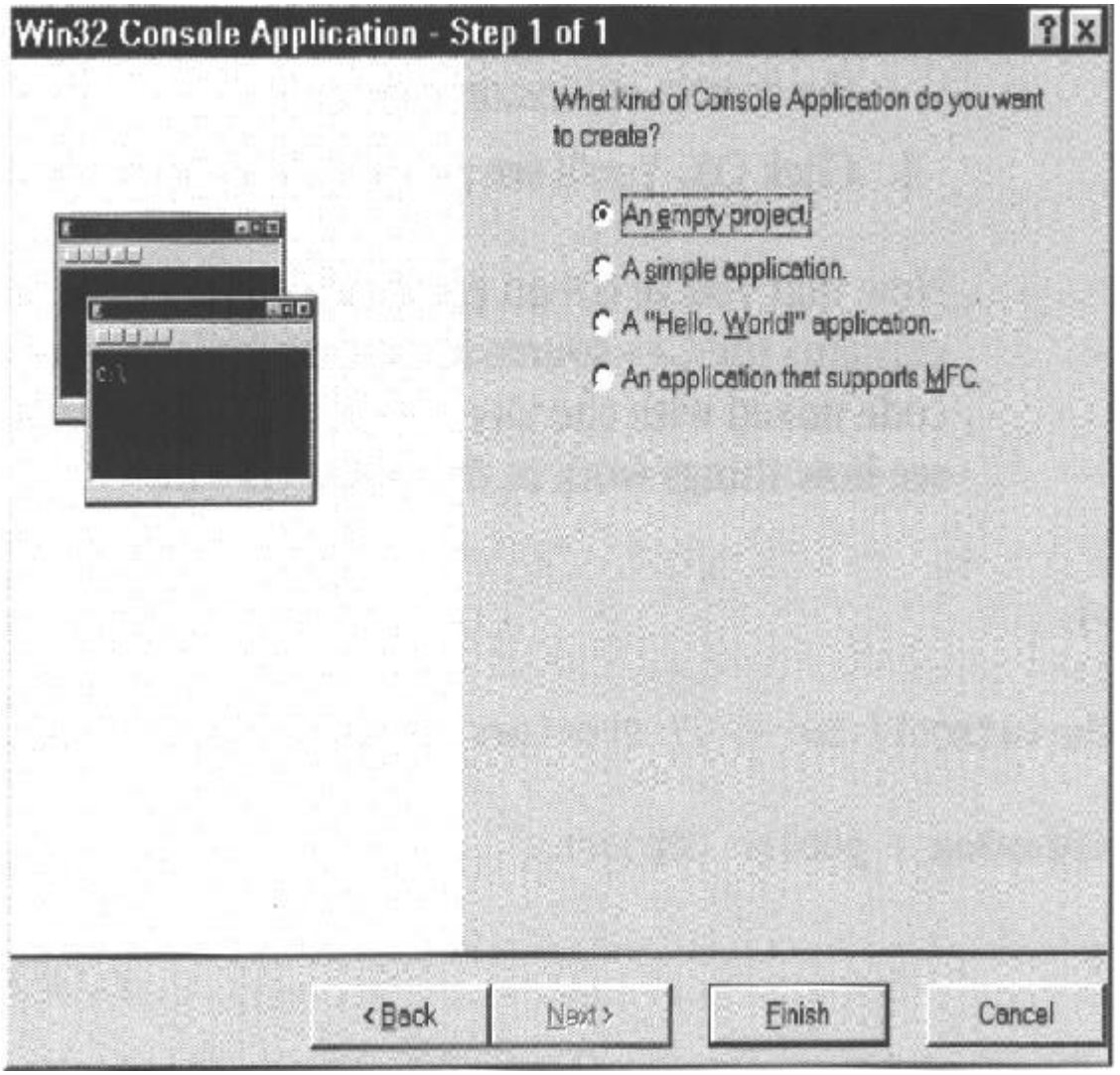
1. 如果你还没有启动 Visual C++，那么启动 Visual C++。
2. 使用 File|New 命令显示如下图所示的 New 对话框。注意，我已经选择了 Projects 选项卡并加亮了要在本例中使用的工程类型。



3. **当选择了** Win32 Console Application **后，在** Project Name **域中键入程序名称。这个样本程序使用的名称为** Console**。你或许还需要修改一下** Location **域的内容，那么单击一下该域旁边的浏览按钮，系统会显示** Choose Directory **对话框，在这个对话框中选择应用程序的存放目录。**

4. **单击** OK **按钮。你会看到** Win32 Console Application - Step 1 of 1 **对话框，如下图所示。注意，在这个对话框中有几个应用程序类型供选择。这也是** Visual C++ 6.0 **的一个新特性。以前版本的** Visual C++**简单地创建一个空的工程。现在**

到了选择要创建哪种类型工程的时候了（即使空工程也要进行选择）。



5. （必要时）选择 An Empty Project，然后单击 Finish。你会看到一个 New Project Information 对话框，它告诉你都选择了哪些选项。

6. 单击 OK 按钮创建示例程序。

在该工程真正能够运行之前，还需要完成其它一些步骤。该工程需要 MFC 类。当你把对该类的支持加入到上面的应用程序向导的设置中时，再次使用该

向导把 DOS 应用程序转换到 Windows 下时所需的重写代码的代码量就会显著减少。选择 Project | Settings 命令显示 Project Settings 对话框，选择该对话框的 General 选项卡，并在 Settings For 组合框中选择 All Configurations，在 Microsoft Foundation Classes 组合框中选择 Use MFC in a Shared DLL 选项。单击 OK 完成设置工作。

现在到了给示例程序添加代码的时候了。需要完成的第一件事是向工程中添加一个文件，让我们看看添加过程。

1. 使用 File | New 命令打开 New 对话框，选择 File 选项卡，其中显示了文件类型的完整列表，包括 Resource Template 以及像 Icon File 这样的各种图像文件。

2. 加亮 C++ Source File 选项。由于我们要在样本代码中添加一些类，因此要选择 C++源文件选项。

3. 在 File Name 域中键入 Console（Visual C++会自动添加上正确的文件扩展名）。

4. 单击 OK。系统显示一个空的 C++源文件。

现在我们有了一个要使用的空文件，该添加一些代码了。程序列表 2.1 显示了该示例的 C++源程序。注意，其中它既直接包括了一些 C 代码，也包括了一些 C++代码。我这样做的目的就是为了让读者看清楚在这种环境下系统是如何工作的。

程序列表 2.1

```
#include    <afxcoll.h>        //Provides Access to MFC functions
Class CDrawBox : public CObject
{
public :
        // Draws the box.
        void DoDraw(char * string)
};

void CDrawBox :: DoDraw(char * cValue)
{
        int             iCount;         //Loop counter
        int             iSpace;       //Amount of spaces to add for string.
        // Draw the top of the box
        fprintf(stdout,"\311");
        for (iCount = 1;  iCount <= 78 ; iCount ++)
        {
                fprintf(stdout,"\315");
        }
        fprintf(stdout,"\273");
```

```
// Figure out the center of the string, then display it
// with the box sides.
iSpace = (80 - strlen(cValue)) / 2;
fprintf(stdout,"\272");
for (iCount = 1;  iCount <=  iSpaces ; iCount ++)
{
      fprintf(stdout,"   ");
}
fprintf(stdout, "%s", cValue);
// Compensate for odd sized strings, then complete the side.
if ((strlen(cValue) % 2) == 1)
{
      iSpaces --;
}
for (iCount = 1;  iCount <=  iSpaces ; iCount ++)
{
      fprintf(stdout, "   ");
}
fprintf(stdout,"\272");

// Draw the bottom of the box
```

```
        fprintf(stdout, "\310");
        for (iCount = 1;  iCount <= 78 ; iCount ++)
        {
                fprintf(stdout, "\315");
        }
        fprintf(stdout, "\274\n");
}

int main(int    argc, char  **    argv)
{
        char *          cName;       // Name of person typed at command line.
        char *          cLocale;     // Program execution location.
        CTime           oMyTime;    // A time object.
        CString         cDate;       // String used to hold time and date.
        CDrawBox    oMyDraw;    // Special text display.

        // See if we have enough command line arguments.
        if ( argc != 2)
        {
                fprintf(stderr, "Type the program name followed by your name.\n");
                return 1;
```

```
    }

    // Get the command line arguments
    cLocale = argv[0];
    cName = argv[1];

    // Get the current time and put it in a string.
    oMyTime = CTime::GetCurrentTime();
    cDate = oMyTime.Format( "%A, %B %d,%Y" );
    //Display everything we've collected.
    fprintf(stdout, "Hello %s\n\n", cName);
    fprintf(stdout, "Program is executing from: \n%s\n\n", cLocale);
    fprintf(stdout, "The date is: %s\n", cDate);

    // Use our class to draw a box around some text.
    oMyDraw.DoDraw("It's a box!");

    return 0;
}
```

如你所见，在这个示例中我展示了四个基本技巧。第一个技巧是对命令行参数适宜个数的检查，如果不满足所需个数的话，就在 stderr **设备上显示一条**

出错信息，然后返回出错代码并退出应用程序。你可以在 DOS 命令行中检测这个错误代码，但对 Windows 来说根本不起作用。一旦代码检测到有足够的命令行参数，它就把这些参数放入一对变量中，用于随后的显示。

到现在为止你看到的还都是 DOS 应用程序的代码。代码中使用的第二个技巧是得到当前时间，它使用了一个 MFC 调用来完成这项任务。那么，如何才能够在控制台应用程序中访问 MFC 函数呢？正如你所看到的，我在代码的一开始就包含 AFXCOLL.H 文件，这个文件包含了控制台应用程序中调用的有限个 MFC 函数的所有定义和类定义。

不要以为你可以任意使用 MFC 调用。例如，不能创建 CDialog 对象并实际使用该对象。如果这样做的话，即使你让代码通过了编译，也可能会产生下述两种情况：或者程序以运行时错误而终止运行，或者应用程序忽略与对话框有关的代码。

技巧 在确定哪些 MFC 类可以使用时的一条重要原则是看该类是否操作图像元素。如果某个类操作图像元素的话，你就不能在控制台应用程序中使用该类。另外，你还会发现某些系统调用不再有效，并且当系统调用涉及安全问题和磁盘访问时，你还必须小心仔细地进行实验。如果存在疑问的话，那么就只好使用 AFXCOLL.H（或其它任何相关的头文件，比如 AFX.H）中列出的调用，而不使用其它调用。在任何控制台应用程序中你都可以安全地使用 AFXCOLL.H 文件中列出的所有调用。

现在我们已经得到了一些要显示的数据，随后的代码把这些数据发送到 stdout（标准输出）设备上，这就是我要向读者展示的第三个技巧。在绝大多数

t（标准输出）设备上，这就是我要向读者展示的第三个技巧。在绝大多数情况下，stdout 代表了显示，当然，如果需要的话，你也可以轻易地把输出发送到其它地方。这里的要点是使用了与以前使用格式化方式相同的格式化方式。这样，你也可以在代码中使用 fprintf()函数调用的地方使用简单一些的 fprint()函数调用。

在 main()函数中还有一个调用没有讲到，我们把数据发送到一个名称为 CDrawBox 的类中，这是我要向读者展示的第四个技巧。这个类要完成的任务就是在一个文本框中让文本居中对齐（使用 ASCII 字符集的高位字符）。你或许在许多 DOS 应用程序中见到过这种做法。这里的想法是从 MFC CObject 类中导出一个新类，然后在控制台应用程序中使用该新类。同样地，在你自己创建的应用程序也可以使用类似的技巧。正如前面我曾经介绍过的那样，当你把 DOS 应用程序迁移到 Windows 环境时，你会有一种向应用程序中添加更多功能的强烈愿望。添加功能意义的大小依赖于应用程序迁移时所耗费的时间长短以及增强功能对整个应用程序的相对价值的大小。

这个控制台应用程序示例可以开始运行了。然而，与你写过的许多 DOS 应用程序相似，为了便于测试，我们需要一个批处理文件。程序列表 2.2 是本例使用的批处理文件的源代码，它完成的任务是调用程序、测试是否出错、如果出错时反馈一条出错信息。前面你已经看到了该应用程序的输出，这里就不再重复了。

程序列表 2.2

```
@ ECHO  OFF
```

```
CONSOLE
IF ERRORLEVEL==1 ECHO IT'S BAD
CONSOLE JOHN
IF ERRORLEVEL==1 ECHO IT'S BAD
@ ECHO  ON
```

2.3 编写基于对话框的应用程序

基于对话框的应用程序十分常用于像实用程序、系统监控程序甚至向导这样的小型任务中。绝大多数情况下，它们专门用于把复杂程度降到最低限度的应用程序中。实际上，可以安全地假定基于对话框的应用程序只使用很少的控件。

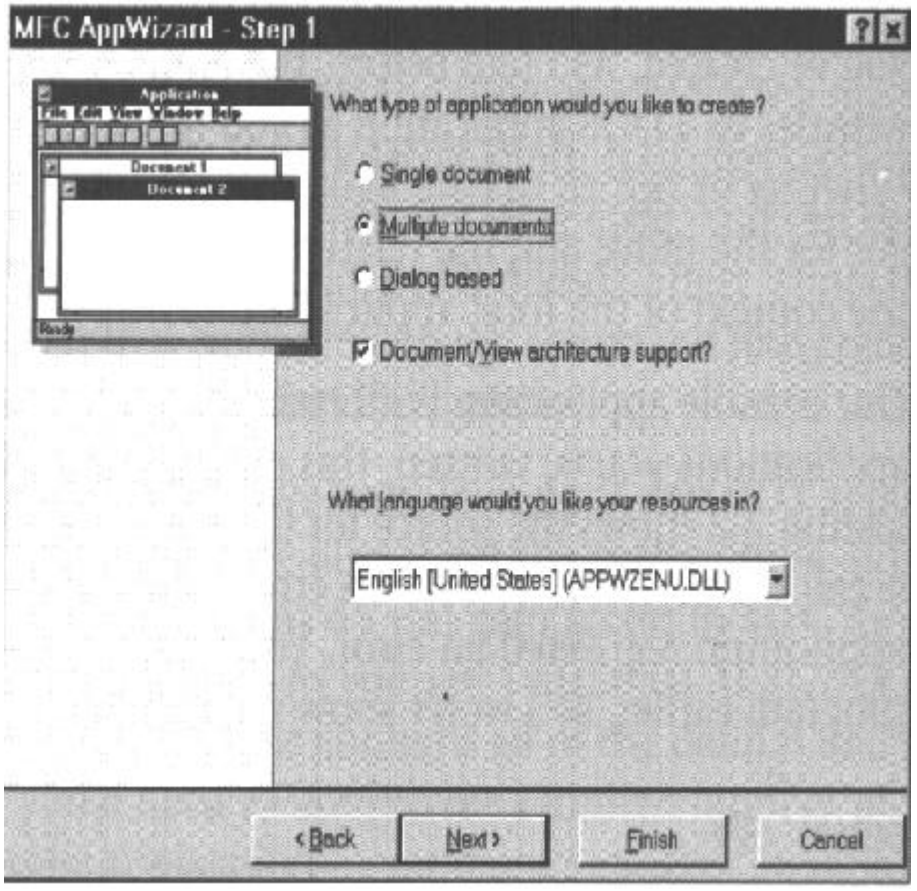
我们要看一看另外一个简单示例。在这个示例程序中，我们要组合使用 ActiveX 控件和内置部件，以把所需的实际代码量减少到最低限度。下面的步骤将帮助你建立一个空的结构，随后再填充代码。

1. 打开 Visual C++（如果还没有打开的话）。

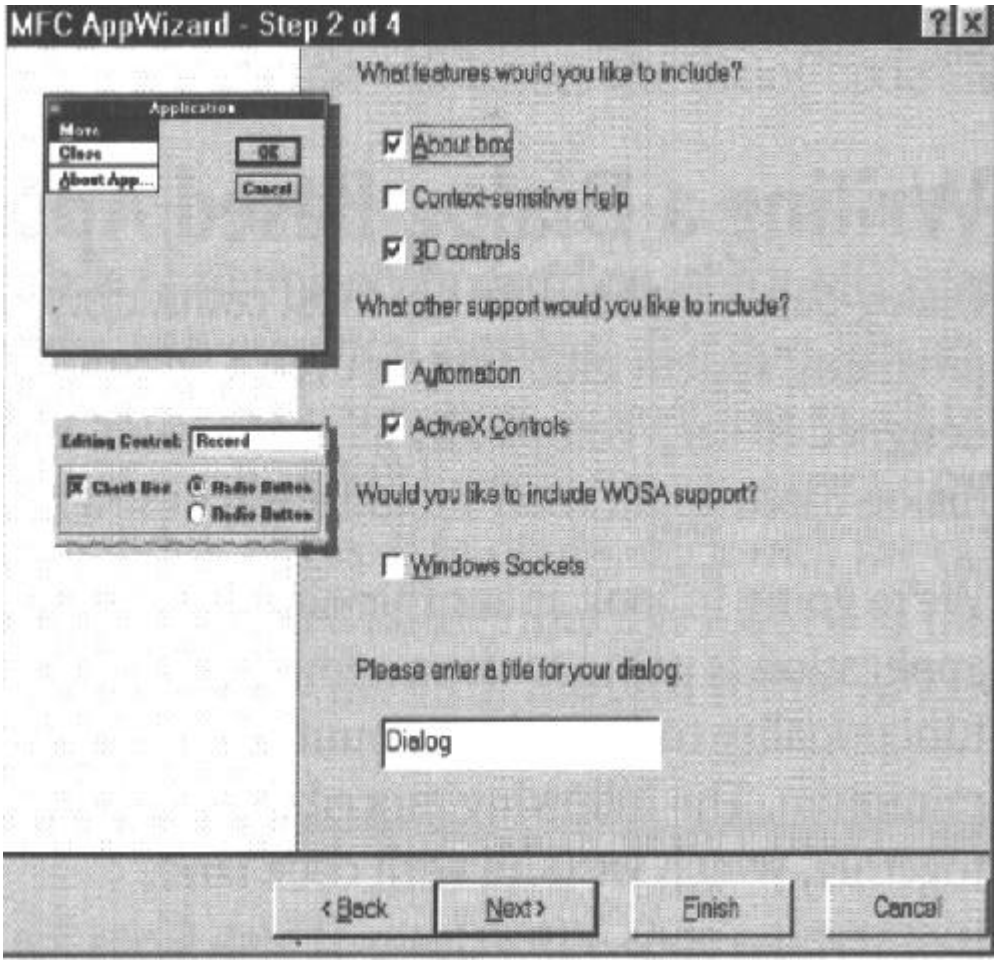
2. 使用 File | New 命令打开 New 对话框，然后选择 Project 选项卡。本例中，我们将在工程类型列表中选择 MFC AppWizard（EXE）工程类型。

3. 在 Project Name 域中键入应用程序的名称。本样本程序使用的名称为Dialog，但你也可以使用任何自己喜欢的名称。必要时修改 Location 域的值（单击 Location 域旁边的按钮进行选择）。

4. **单击** OK **按钮，你会看到如下图所示的** MFC AppWizard - Step 1 **对话框。**



5. **选择** Dialog Based **选项按钮，然后单击** Next，**系统显示如下图所示的** MFC AppWizard - Step 2 of 4 **对话框。**

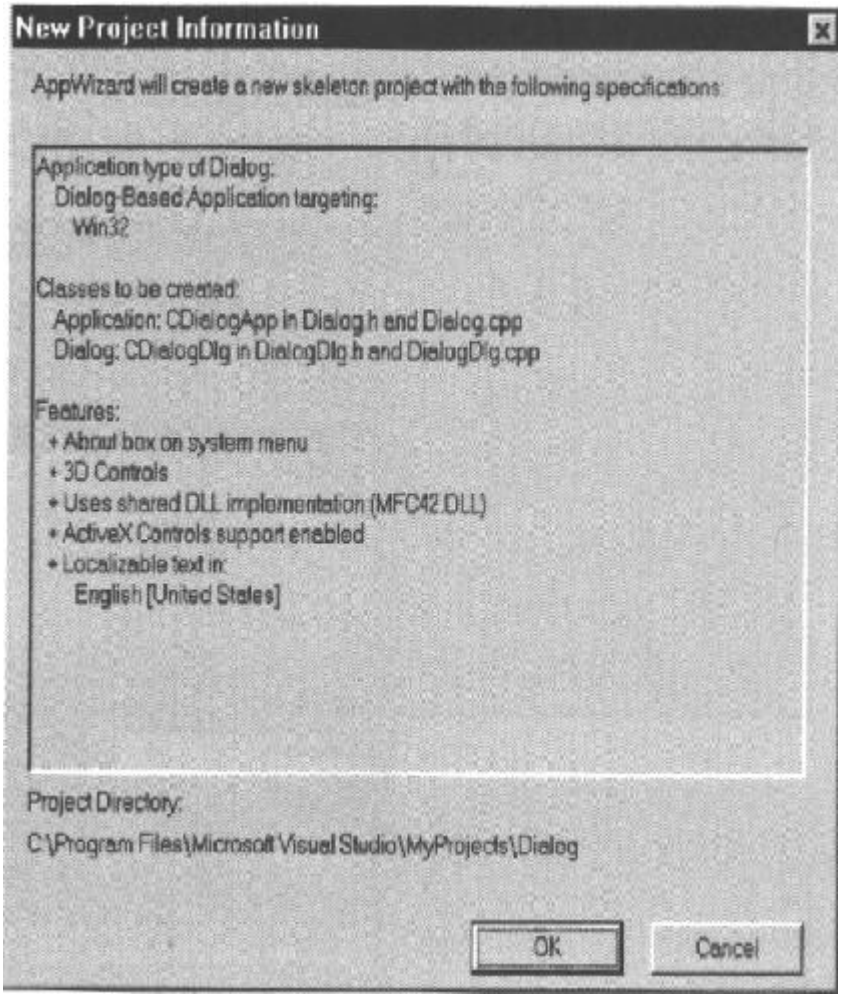


6. **在** Please Enter a Title for Your Dialog **域中键入** Sample Dialog Application**。注意，在这个对话框中可以给应用程序增加一些其它特性，比如上下文相关帮助等。**Automation **复选框允许把** OLE **自动化功能添加到应用程序中。你也可以在这个对话框中对应用程序增加** Windows Sockets **支持——该支持允许应用程序通过** TCP/IP **网络进行通信。**

技巧 把 MFC 静态链接到应用程序中可以减少发布应用程序时必须分发的文件的数量。实际上，如果愿意的话，你可以只向其他人提供应用程序的可执行文件。这种方式也提高了应用程序在各个所安装机器上正常运行的机会，原因在于应用程序总是访问设计应用程序时使用的 MFC 版本。静态链接的缺点是应用程序变得更加庞大，而且在加载时占用更多的内存。另外，无论什么时候，当你想给应用程序增加新的特性时，都必须重新链接应用程序，不久这种方式就会让人感到厌烦。

7. **单击** Next，**然后再单击** Next。**我们不需要改变这两个对话框中的设置，但应该知道它们的作用。前一个对话框让你选择是否要在代码中添加注释。你还可以选择** MFC **是静态还是动态链接到应用程序中。**

8. **单击** Finish。Visual C++**显示如下图所示的** New Project Information **对话框。这是你确保各种设置都正确无误的最后一次机会。不正确的设置选择将延长工程的开发时间而不是缩短开发时间。**

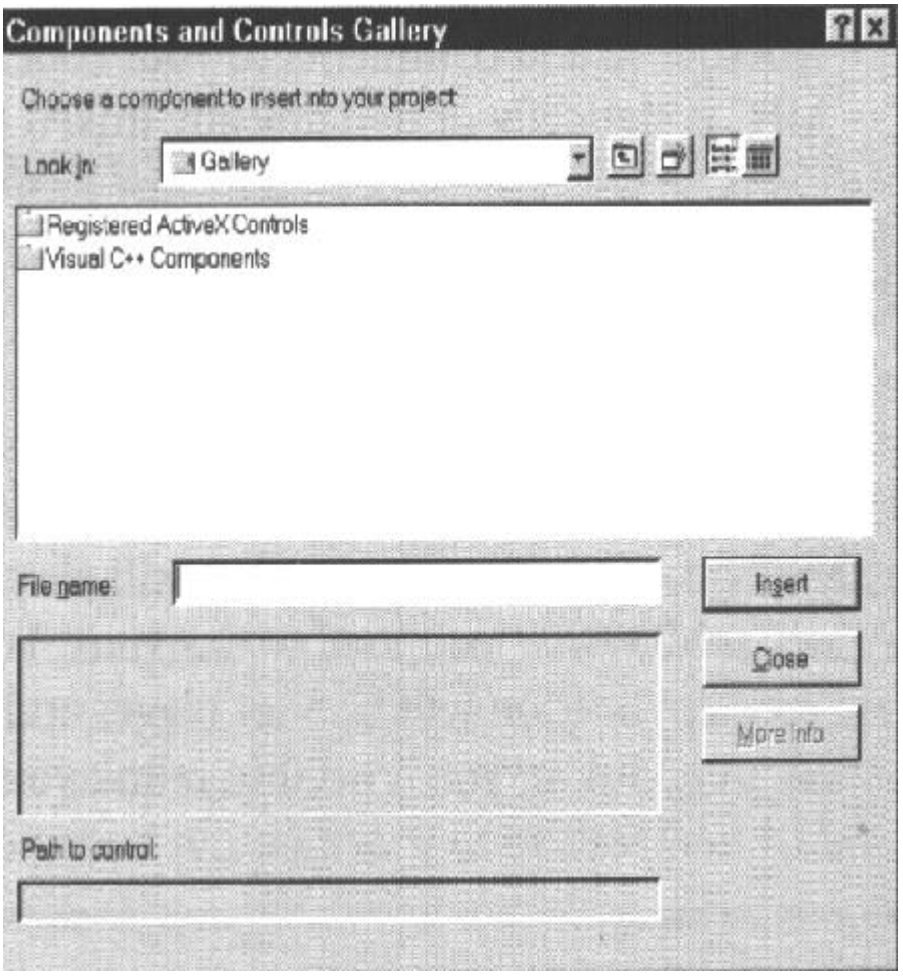


9. **单击** OK。MFC AppWizard **将生成应用程序的框架。**

现在我们已经得到了一个可以使用的基本程序框架，到了实际开始编写示例程序的时候了。由于这个程序要使用 Microsoft Calendar **控件，所以我们要做的第一件事就是安装所需的** ActiveX **控件。使用** Project | Add to Project | Components and Controls **命令显示如下图所示的** Components and Controls Gallery **对话框。**

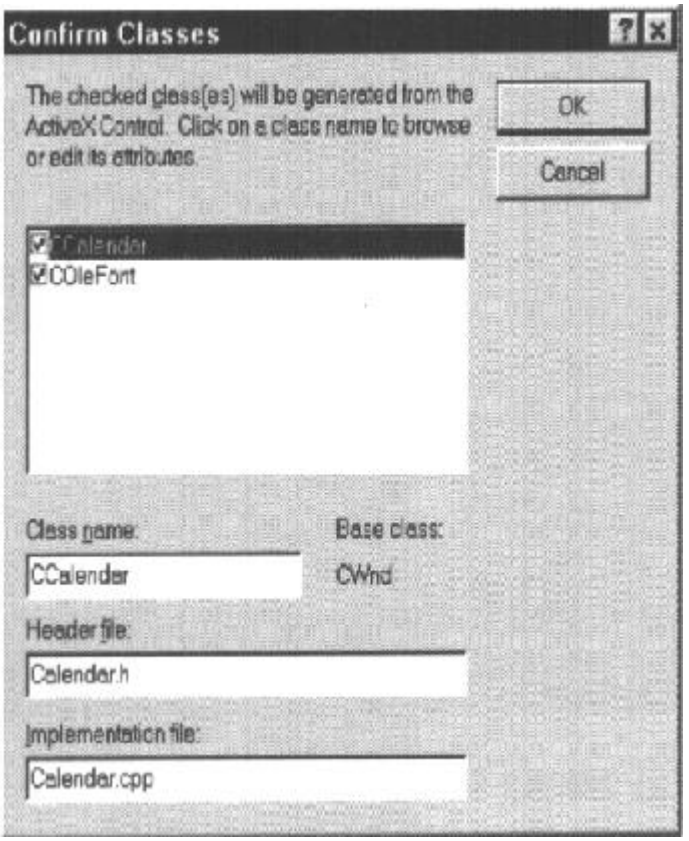
注　确保已经安装了 Visual C++携带的 OCX 控件，否则可能就没有可用的

Microsoft Calendar 控件。



双击 Registered ActiveX Controls **文件夹，你就会看到机器中已经注册的一系列控件。找到** Calendar Control 8.0 **列表项（这是本版** Visual C++**携带的控件之一），然后单击** Insert。Visual C++**将显示一个对话框询问是否真要把这个控件插入到工程中。单击** OK。Visual C++**将添加某些包装类到应用程序中，以便为其提供该** ActiveX **控件。**Visual C++**在如下图所示的** Confirm Classes **对话框中显示这些类的类名。**

单击 OK 接受缺省的 ActiveX 控件类设置。现在我们有了一个可以使用的新控件，它被添加到了工具箱中。单击 Close 按钮关闭 Components and Controls Gallery 对话框。



现在到了设计对话框的时候了。图 2.1 显示了设计好的对话框。该对话框的当前大小为 300×200，我把 Calendar 控件的大小调整为 230×186，这样该控件上的数字就看得清楚些了。请注意，虽然该应用程序现在并没有运行，但 Calendar 控件依然立即显示当前日期，这是由于在把 ActiveX 控件放置到对话框上时，该控件自动激活了自身。如果在你把某个控件放置到对话框上时并没有发生其它反应，或许该控件没有正确地安装。显然，需要检查控件的文档以确保控件

按所期待的方式工作。

技巧 在屏幕右部状态条第二个方框中总能看到当前控件的大小。紧挨着该方框的左边方框中显示了所选控件相对于显示区左上角的位置。利用这两个方框可以十分精确地控制控件在对话框上的大小和位置。

如果现在编译和运行这个应用程序，你会发现它只是个半成品。日历会让你选择新的日期，单击 OK 按钮后对话框也会消失，但也只是如此，该程序现在还没有更多的其它功能。

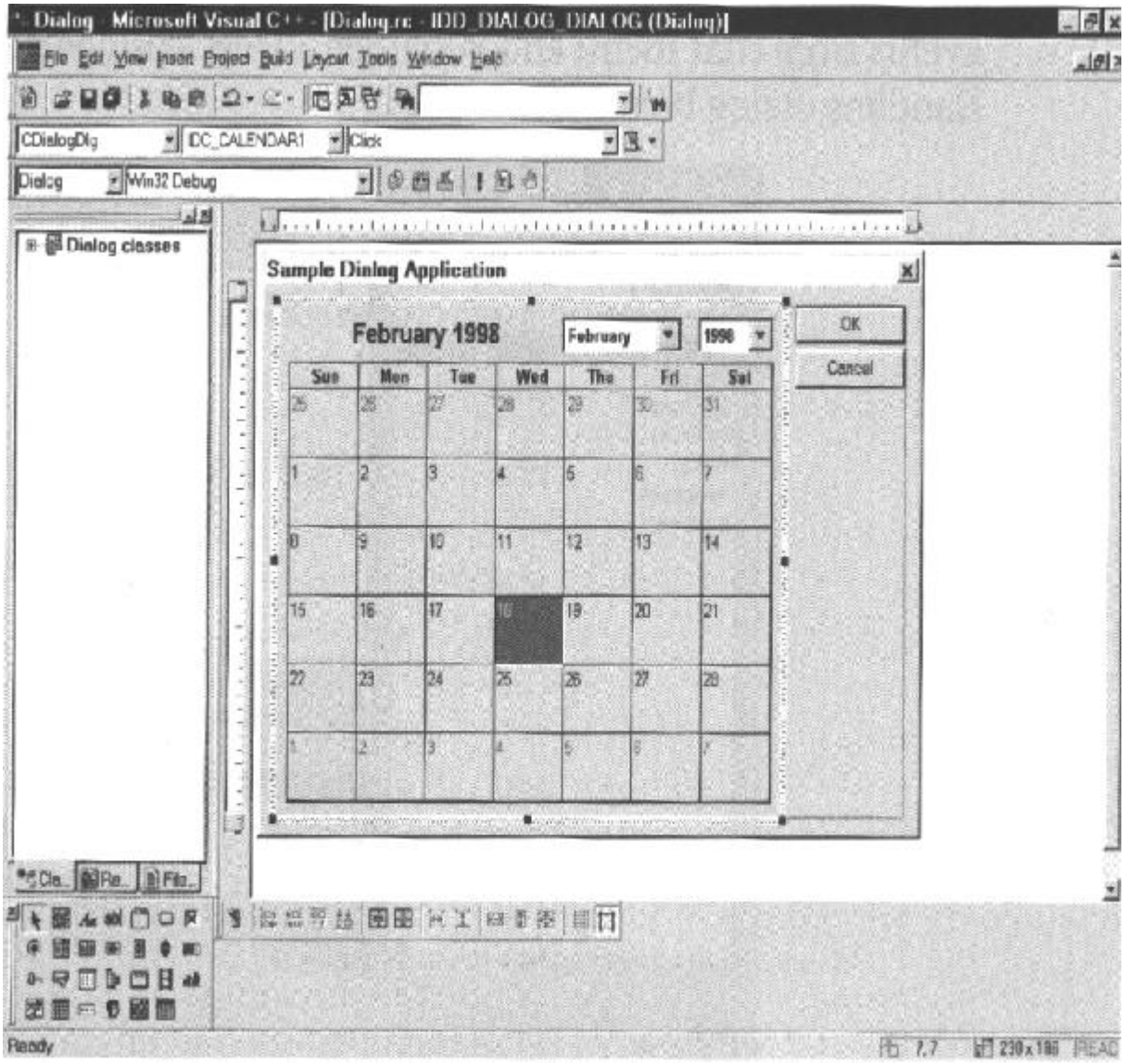
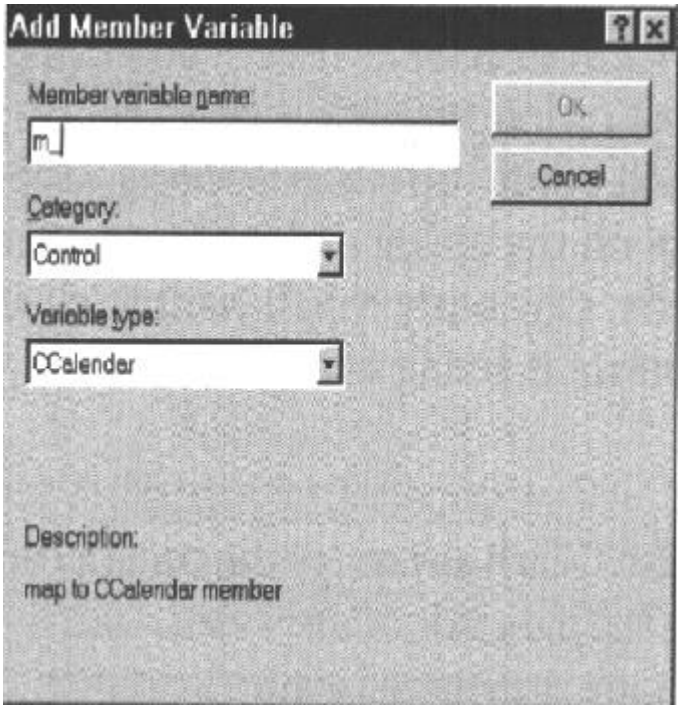


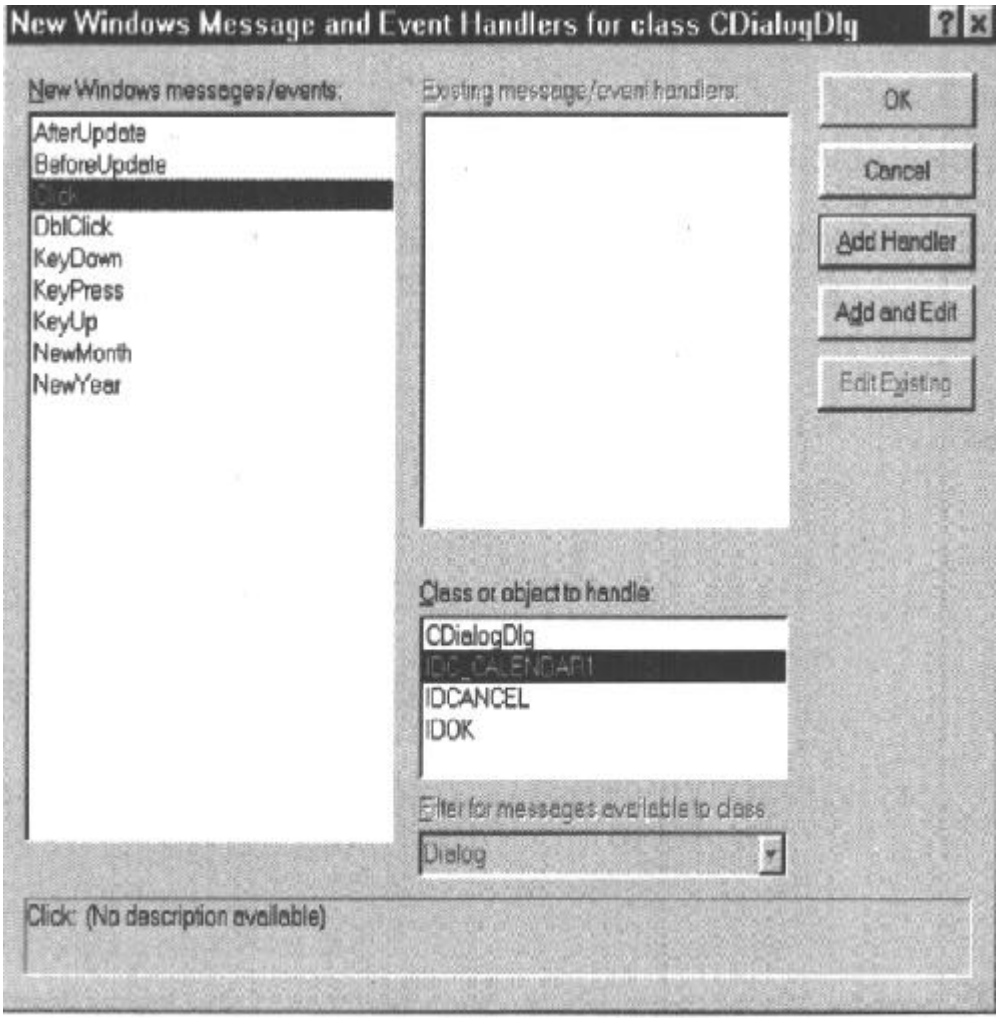
图 2.1　**加入控件后，基于对话框的示例程序开始成型了**

在给 Calendar **控件添加代码之前，我们还要为它创建一个成员变量。创建成员变量是件简单的事，按住** CTRL **键双击** Calendar **控件，系统显示如下图所示的** Add Member Variable **对话框。**



由于该控件是应用程序中使用的第一个 Calendar **控件，因此在** m_**（不要删除** m_**）后键入** Calendar1**，单击** OK **创建该变量。**

下一步要完成的任务是检测用户是否改变了该控件。右击 Calendar **控件，你会看到一个上下文菜单，从中选择** Events**，系统显示如下图所示的** New Windows Message and Event Handlers **对话框。**



该对话框包含了控件可以监视的所有事件的完整列表。程序员在设计该控件时设置了这些事件。在本例的情况下，我们要监视 DblClick **事件。加亮** DblClick **事件列表项，然后单击** Add and Edit **按钮，**Visual C++**显示** Add Member Function **对话框。单击** OK **接受缺省的函数名，你会看到一个空的函数体。现在我们要完成的任务就是加些代码让函数正常工作。程序列表** 2.3 **显示了所需的程序代码。**

程序列表 2.3

```
void CDialogDLg::OnDblClickCalendar1()
{
    CString     cSelectedDate;      // Date selected by user.
    Char *      cDay  =  "    ";      // Selected day.
    Char *      cYear  =  "    ";      // Selected year.

    // Get day from calendar control.
    itoa (m_Calendar1.GetDay(), cDay,10);
    cSelectedDate = cDay;

    // Get month from calendar control.
    switch (m_Calendar1.GetMonth())
    {
    case 1:
        cSelectedDate = cSelectedDate + " January ";
        break;
    case 2:
        cSelectedDate = cSelectedDate + " February ";
        break;
    case 3:
```

```
        cSelectedDate = cSelectedDate + " March ";
        break;
case 4:
        cSelectedDate = cSelectedDate + " April ";
        break;
case 5:
        cSelectedDate = cSelectedDate + " May ";
        break;
case 6:
        cSelectedDate = cSelectedDate + " June ";
        break;
case 7:
        cSelectedDate = cSelectedDate + " July ";
        break;
case 8:
        cSelectedDate = cSelectedDate + " August ";
        break;
case 9:
        cSelectedDate = cSelectedDate + " September ";
        break;
case 10:
```

```
            cSelectedDate = cSelectedDate + " October ";
            break;
        case 11:
            cSelectedDate = cSelectedDate + " November ";
            break;
        case 12:
            cSelectedDate = cSelectedDate + " December ";
            break;
        }

        // Get the year.
        itoa(m_Calendar1.GetYear(), cYear,10);
        cSelectedDate = cSelectedDate + cYear;

        //Display the date.
        AfxMessageBox("You double-clicked on:" + cSelectedDate , MB_OK |
            MB_INFORMATION, 0);
}
```

现在，你可以运行该应用程序了，它也确实会完成一些任务。编译并运行这个应用程序。双击任意日期，程序显示如图 2.2 **所示的对话框，该对话框以易于阅读的方式显示了你在** Calendar **控件中所选的日期。虽然这个功能现在看**

起来并没有多大用处，但你有多种方法轻易地扩展这个实用程序的功能。例如，你可以在每次双击日期时显示一个弹出式的小型记事本，在记事本中记录每天的日程安排。毫无疑问，这个程序不会取代你的合同管理器，但作为一个笔记本它确实干得不错——特别是在空间有限的膝上机和你并不想让大型应用程序用尽电池的情况下。

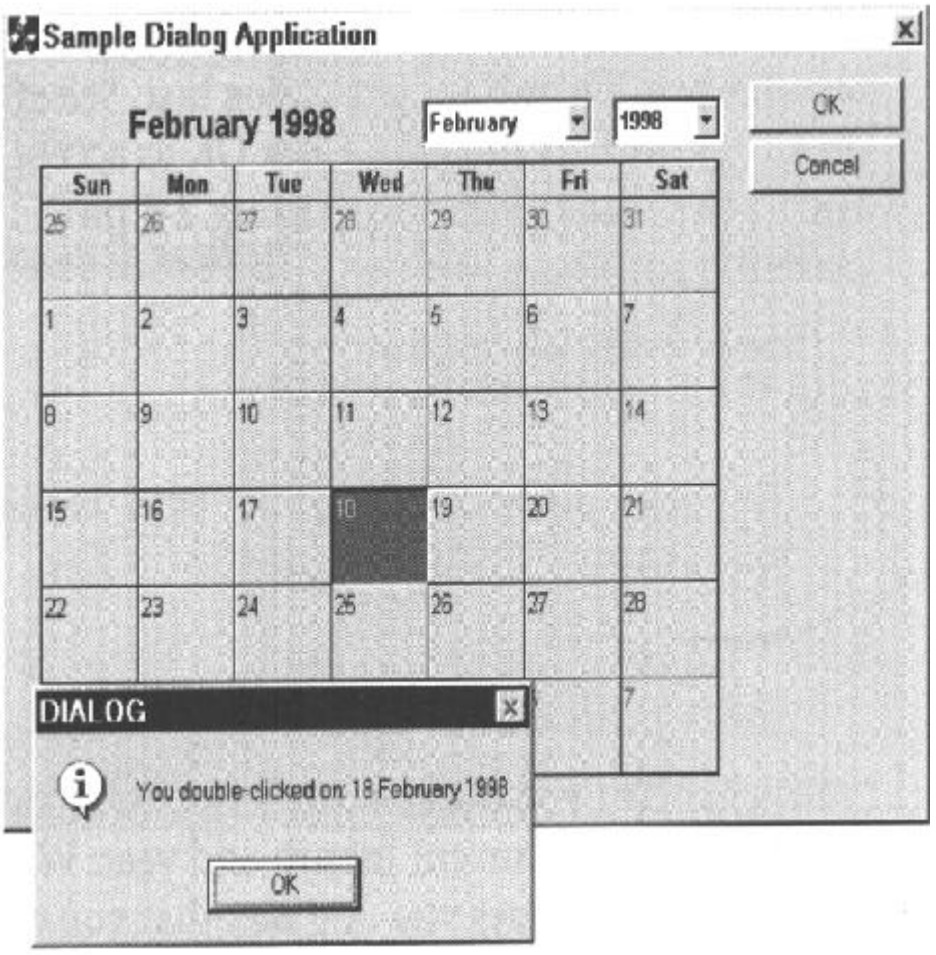


图 2.2　样本对话框应用程序显示一个对话框告诉用户它所选择的日期

Calendar 控件提供了不少的功能——绝大部分本章中都没有涉及。再做一个把日期设置回当天日期的按钮简单方便，这样，当你打电话离开一会儿时就可以用了。添加这样的按钮一点也不难，图 2.3 显示了我已经添加到现存工程中的按钮。它的 ID 为 IDC_RESET_DATE，标题为 Reset Date。如果把该按钮放置在 243,39 并把它调整到 50×40 的大小时，对话框看上去更舒服些。

既然我们有了按钮，那就再添加几行代码吧。右击该按钮并从上下文菜单中选择 Events，系统显示 New Windows Message and Event Handlers 对话框，选择 BN_CLICKED 事件，然后单击 Add and Edit，单击 OK 接受缺省的过程名，你会看到一个空的过程。程序列表 2.4 提供了要在该过程中添加的代码。

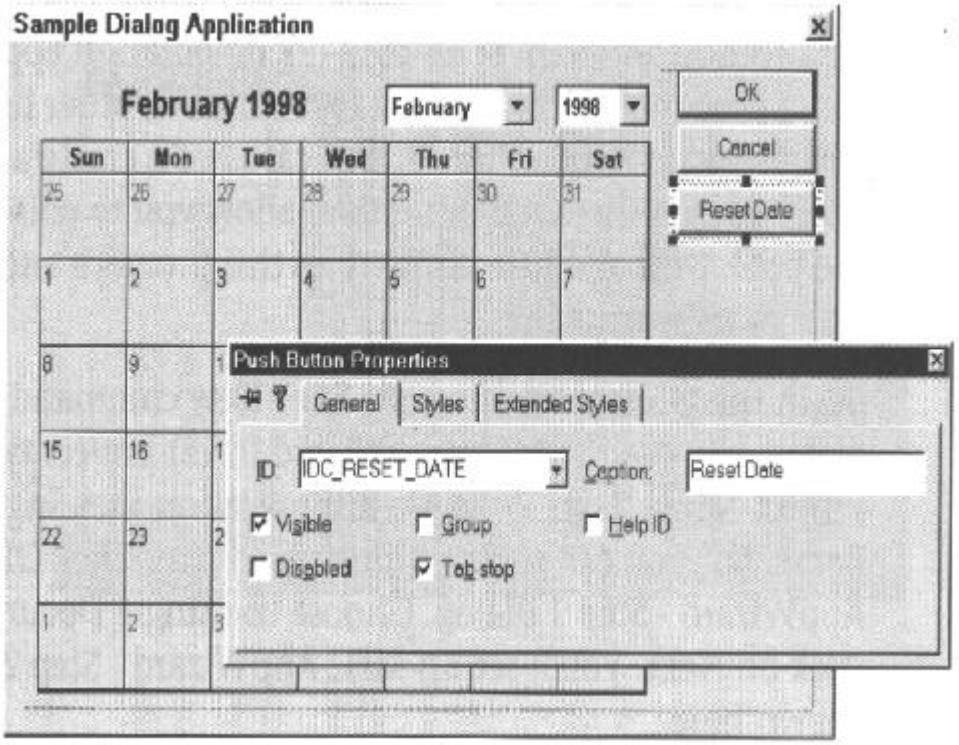


图 2.3　提供个按钮在操作了一阵日历后把日期重新设置到当前日期确实很方便

程序列表 2.4

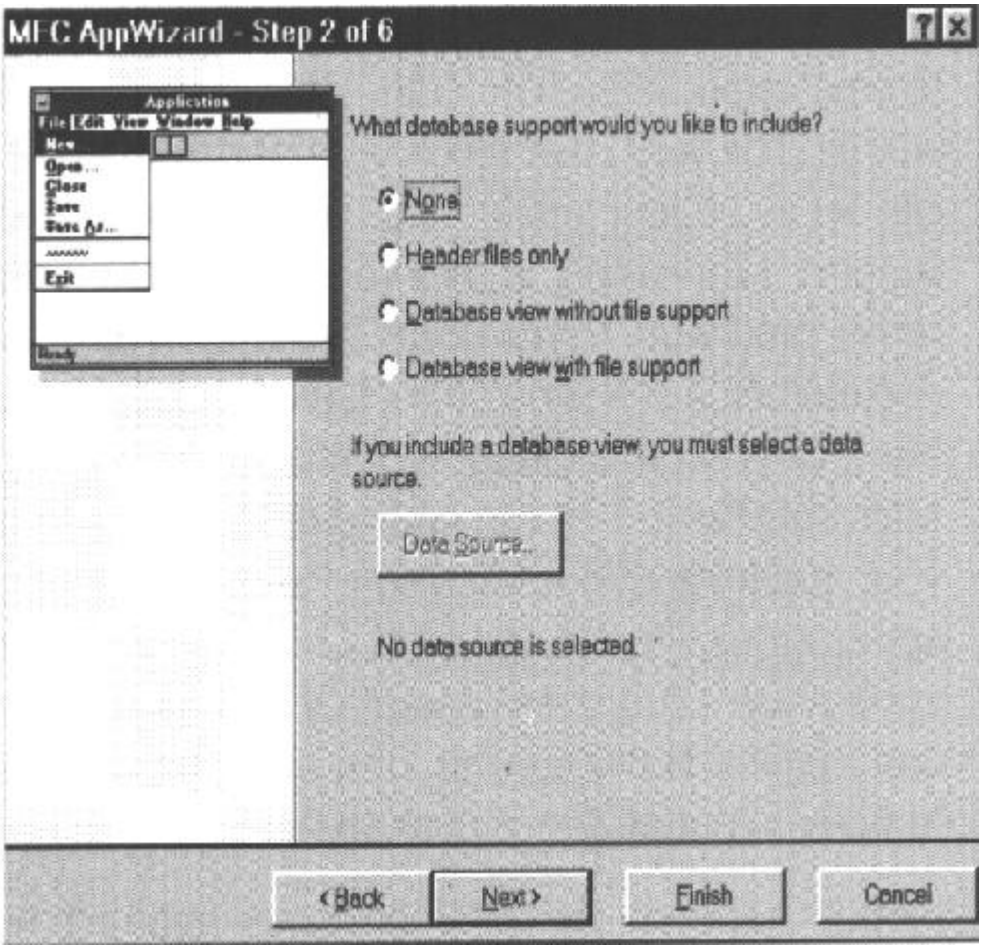
```
void CDialogDLg::OnResetDate()
{
    // Reset selected calendar date to today.
    m_calendar1.Today()
}
```

现在只要你单击一下 Reset Date 按钮，Calendar 控件就会向用户返回当前的月份和年度。虽然这段代码并不复杂，它却展示了在不增加应用程序使用复杂程度的情况下给对话框应用程序添加功能的途径。当然，即使只有少量的这种按钮过上一阵子也会让人厌倦。

2.4 编写单文档应用程序

每个版本的 Microsoft Visual C++ 向导都有轻微的改进，在你集中精力使用某个向导的过程中，设置应用程序时遗漏某些东西也是件毫不奇怪的事情。

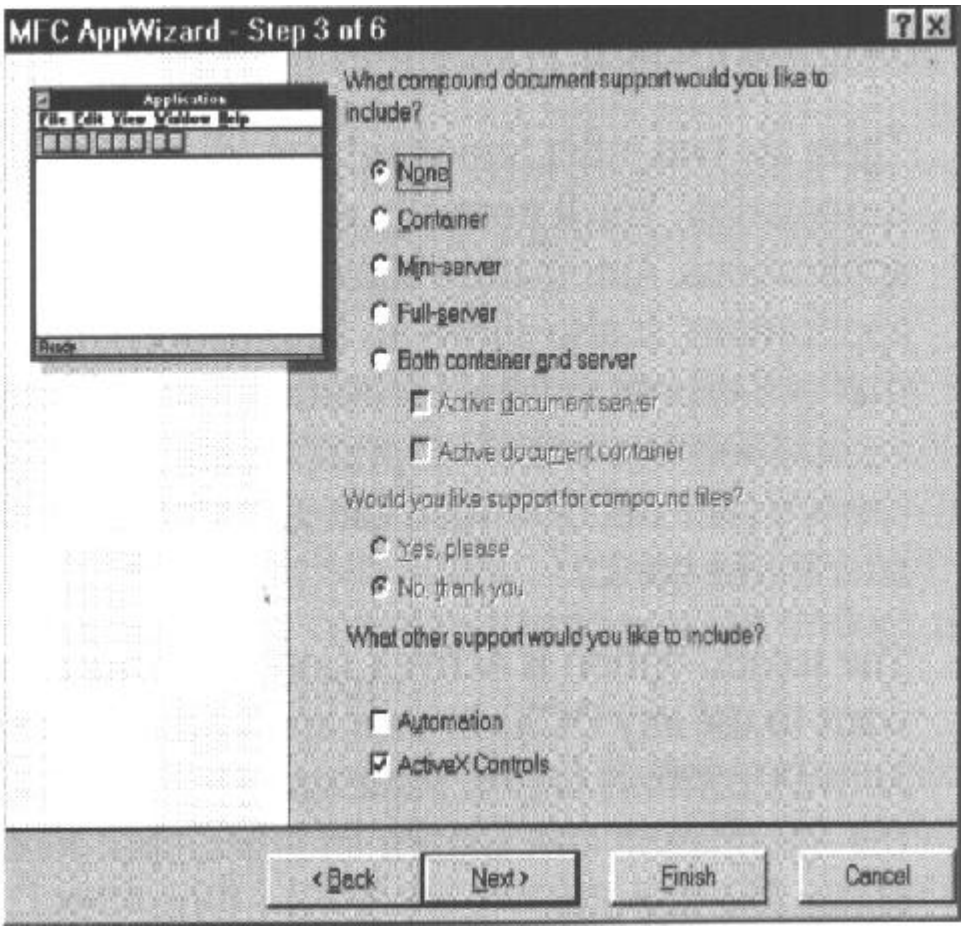
本节研究单文档应用程序——或许你已经用它编写过某个小型的文本编辑器或其它轻型、通用的文档编辑器。本例中，我们将创建一个丰富格式文本编辑器，它不需要做多少工作就可以很好地编辑文本了。在本书后续章节中，我们将给这个程序增加新的功能，使其更加有实用价值。



开始这个工程时，使用 File|New **显示** New **对话框，选择** MFC AppWizard(EXE) **工程类型，在** Project Name **域键入应用名称，示例应用中使用** Sng1_Doc **作为应用程序名称，但你也可以使用与此相似的任何名称。单击** OK，**系统显示** MFC AppWizard - Step 1 **对话框，选择** Single Document **选项，然后单击** Next。**系统显示如下图所示的** MFC AppWizard - Step 2 of 6 **对话框。**

这是需要你做出决策的第一个位置。如果你想在应用程序中增加对数据库的支持，那么需要决定在 MFC AppWizard **中得到哪个层次的支持，在本书的第二部分我们将花些时间详细讨论这个问题，因此这里我就不多讲了。单击** Next，

你会看到如下图所示的 Step 3 **对话框。**

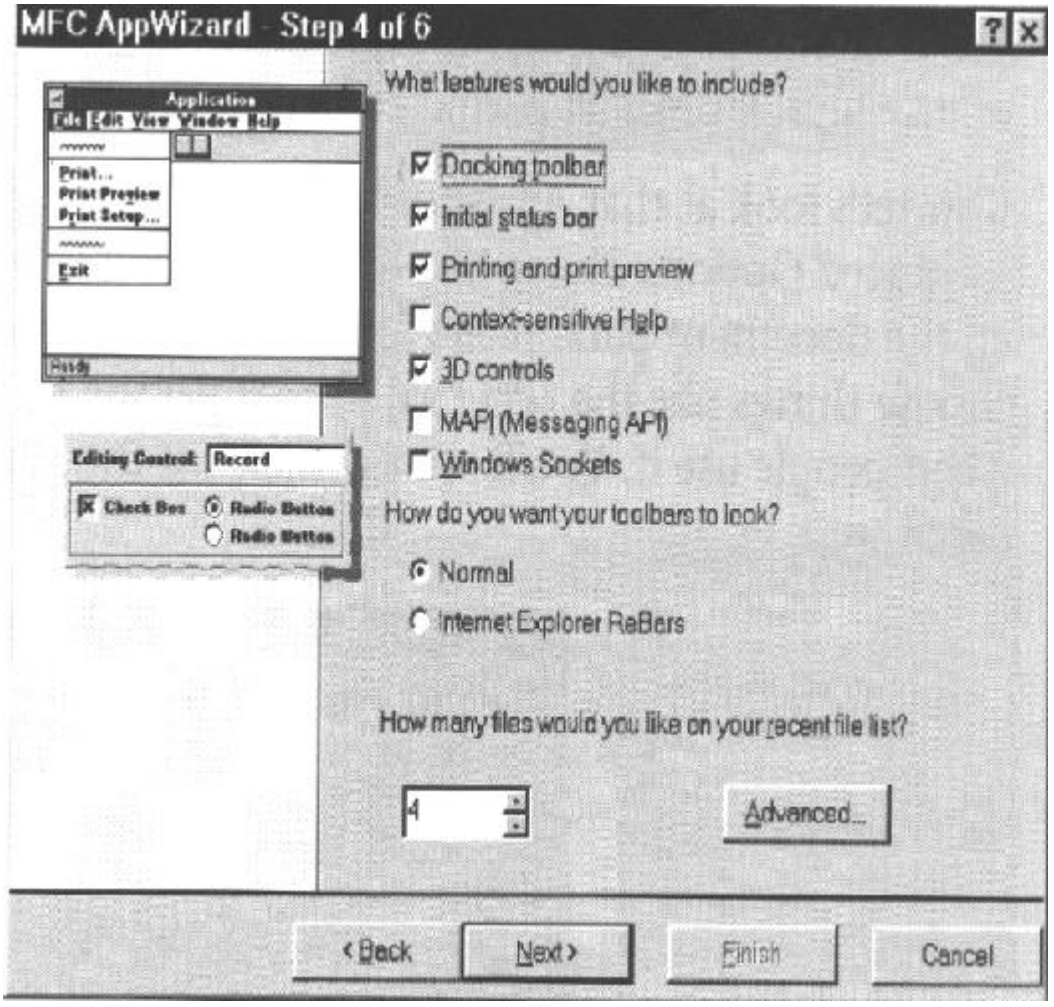


这个对话框的上半部决定了要在应用程序中添加哪个层次的 OLE **支持。添加的支持越多，应用程序也就越大。最基本的** OLE **支持层次是用做容器（**Container**）。容器既可以作为客户，又可以存储链接对象和嵌入对象。下一个** OLE **支持层次是** Mini-server**（小型服务器），它允许创建复合文档，小型服务器不能独立工作。另外，它只能操作嵌入对象。作为服务器使用时，全功能服务器（**Full-Server**）确实具备了** OLE **的全部功能，但它却不能作为容器使用。这样的应用程序用起来与** Microsoft Paint **（画图）十分相似，你可以把** Microsoft Paint **文档嵌入或链接到你的应用程序中，但** Microsoft Paint **本身不能存储其它**

入或链接到你的应用程序中，但 Microsoft Paint 本身不能存储其它应用程序创建的对象。最后，Both Container and Server 选项让你的应用程序具备本地 OLE 支持的所有功能。你既可以把应用程序用做 OLE 服务器，也可以将它用做 OLE 客户。但是，这种应用程序不能用做 Internet 浏览器——原因在于它没有设计成 ActiveX Document 服务器。

注意在最后一个单选钮（Both Container and Server）的下方有两个复选框，如果选中第一个复选框，Visual C++还将增加使你的应用程序用做 ActiveX Document Server（ActiveX 文档服务器）所需的支持。Active Document Server 能够创建和管理 ActiveX 文档。第二个复选框是 Active Document Container，它允许你的应用程序在其框架内包含 ActiveX 文档。本质上说，它允许你在应用程序中嵌入像 Microsoft Word 或 Excel 这样的应用程序生成的文档。现在，我们选择 Both Container and Server 选项并选中 Active Document Server 复选框，在第 11 章中我们将探讨这项选择的细节。

在 Step 3 对话框中还有另外两个重要的复选框。第一个是 Automation，对大型应用程序来说通常要选中这个复选框。Automation 允许你操作其它应用程序创建的对象，也允许自动化（Automation）客户端操作你的应用程序创建的对象。从现在开始，也可以把自动化看成是一种可编程能力，虽然这种说法不甚精确（自动化的完整定义相当长，需要放开讲才能说清楚——在本书的后续章节中我们将讨论这个问题）。



第二个选项是 ActiveX Control**。如果在应用程序中要使用任何** OCX **控件（无论** OCX **控件是否设计成** ActiveX **控件），那么都必须选中这个复选框。确实选中** ActiveX Control **和** Automation **复选框，然后单击** Next**，系统显示如下图所示的** Step 4 **对话框。**

该对话框的上半部分用于选择应用程序中要包含的界面元素的类型。正常情况下，保留 Visual C++**的缺省设置，这样的设置将创建标准的应用程序。如果想为用户提供上下文相关帮助，那么选中** Context-Sensitive Help **复选框。在**

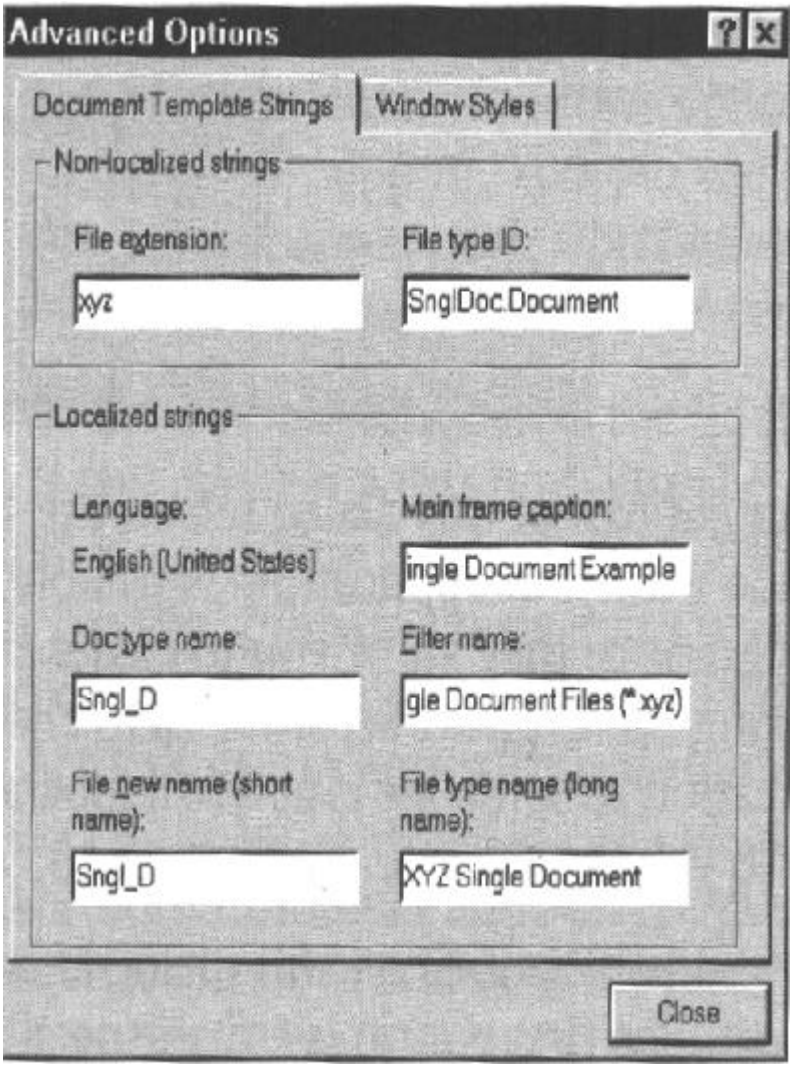
第 15 章中我们将介绍向应用程序中添加上下文相关帮助的方法。

这个对话框中下面的两个选项让你在应用程序中添加通讯支持功能。第一个选项专门用于电子邮件，选中这个选项后应用程序可以发送和接收电子邮件（只要电子邮件应用程序与 MAPI 兼容）。第二个选项让应用程序通过 TCP/IP 网络进行通讯。如果要在你的应用程序中添加 Internet 支持功能，那么选中这个复选框。

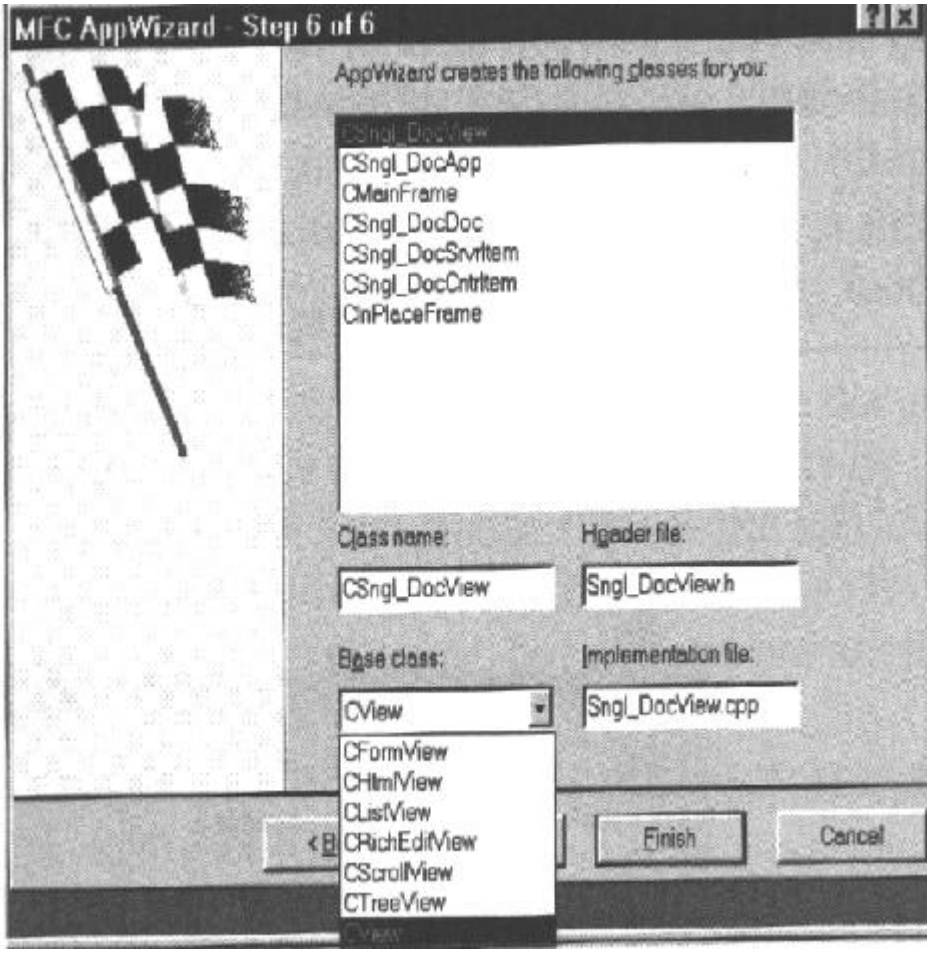
Visual C++包含的新特性让你能够选择工具条的外观。我们既可以选择Normal 使用标准 C++工具条的外观，也可以选择 Internet Explorer 提供的新外观。使用 Internet Explorer ReBars 选项有两个好处：我们既可以在工具条上添加任何 Windows 控件（并不仅限于菜单命令），也可以像命令条那样“漂浮”ReBars。

剩下的两个选项是最近使用文件列表中列出的文件个数和一个标题为Advanced 的神秘按钮。正常情况下，我把最近使用文件列表中列出的文件个数设置为尽可能大的值，原因在于我经常要操作许多文档。在这里给用户提供最大的灵活性并不影响应用程序的大小和速度，因此，将其设置为尽可能大的值并不是什么大是大非的问题。

现在让我们看一看 Advanced 按钮。单击这个按钮，系统显示如下图所示的 Advanced Options 对话框，这个对话框是设置文档参数的地方，也是选择用户使用时显示在标题条中文本这类设置的地方。在标题条中也将显示当前文档的名称。



按上图所示设置 Advanced Options **对话框。完成后的应用程序使用扩展名为** XYZ **的文件，应用程序的标题为“** Single Document Example **”，每次显示** File Open **或** File Save **对话框时，过滤器域显示为“** XYZ Single Document File(*.xyz) **”。最后，当显示该应用程序创建的文档的属性对话框时，对话框将显示该文档的类型为“** XYZ Single Document **”。单击** Close **按钮接受上述设置，单击两次** Next**，系统显示如下图所示的** Step 6 **对话框。**



这是我要向你介绍技巧的另一个地方。如果你没有真正留意的话，就有可能遗漏在文档视图类中可以使用你喜爱的任何基类这一事实。为什么说要重点考虑基类的选择呢？如果接受了 Visual C++**的缺省设置，那么应用程序对文档的任何操作你都必须编写代码。当你要创建一种全新的文档时，上述设置并没有任何不妥，但绝大多数应用程序并非要创建一种全新的文档，而是建立一个与其它应用程序相似、并提供其它应用程序所缺乏特性的应用程序。本例中，我们要创建一个文本编辑器，因此没有必要使用** CView **（应用程序的缺省基类）。**

选择 CEditView 或 CRichEditView 类作为应用程序的基类将省去许多工作，原因在于该应用程序一经设计就已经知道作为简单的字处理程序应该如何完成任务。你甚至不需要添加任何代码就能得到上述功能。

为了展示基类的作用，在示例程序中我们选择 CRichEditView 作为 CSng1_DocView 类的基类。单击 Finish 按钮完成设置。当看到 New Project Information 对话框时，单击 OK，Visual C++为你创建该示例应用程序。

现在生成应用程序并启动运行。你会看到可以在应用程序中输入文本并将其保存到 XYZ 文档中。添加一些代码后，该应用程序就可以读取其它文本和 RTF 文档。当然，也可以以其它方式扩展这个应用程序。例如，由于该应用程序使用了富文本编辑控件，因此可以为其增加文本格式化功能和颜色功能。我们已经添加的 OLE 能力意味着需要时可以在文档中插入图像。已经完成的应用程序只做了很少的工作就具备了许多潜在的功能。

或许现在你还没有注意到，这个特殊的示例程序并没有要你编写一行代码。最终的程序相当完美，是 C 程序员不敢想像的。图 2.4 展示了到现在为止这个应用程序的显示结果，随着本书的展开，我们将进一步改进这个应用程序。

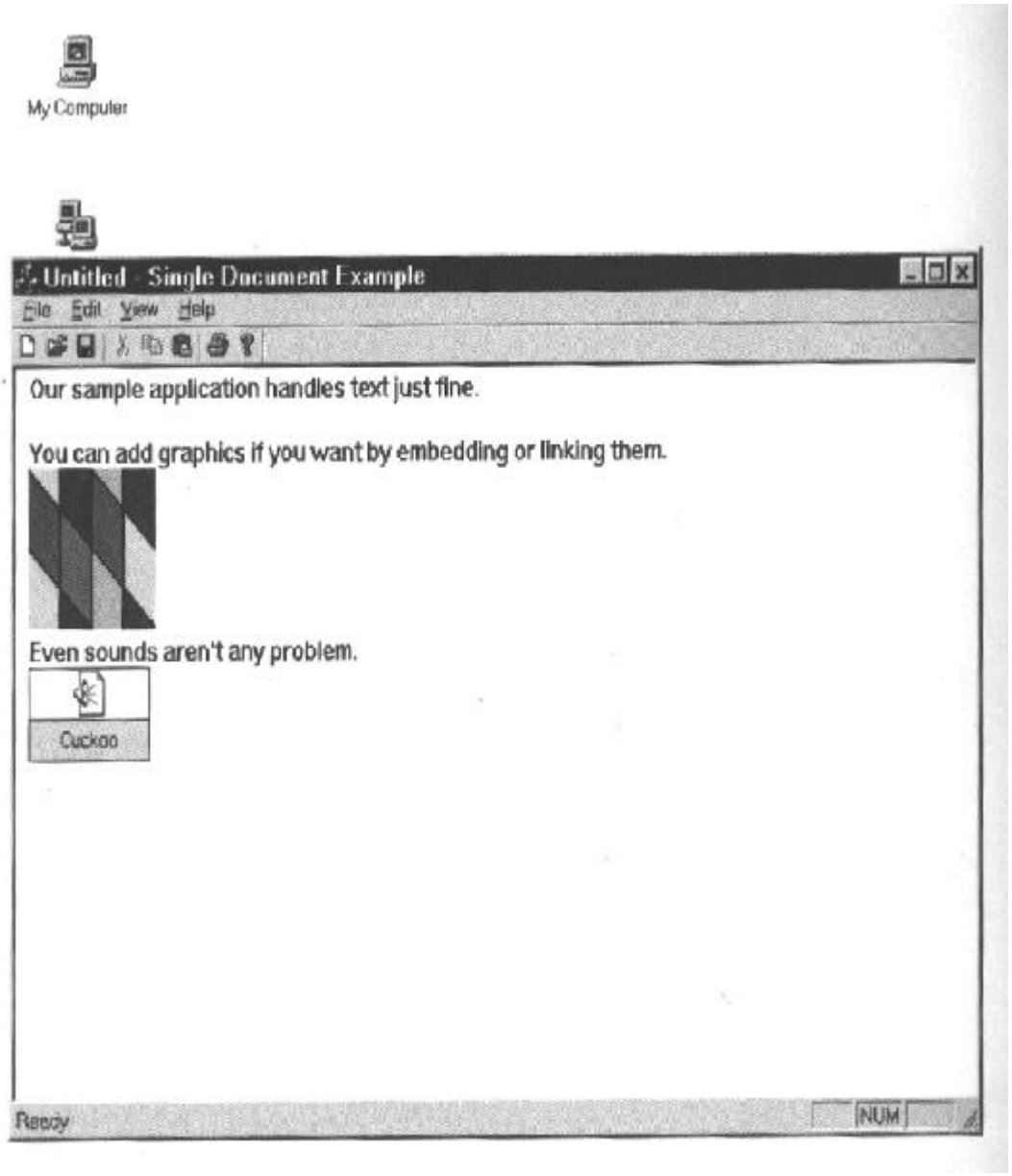


图 2.4　考虑到我们并没有给这个示例应用程序添加任何代码，应该说它运行的够完美了

2.5 编写基于 HTML 文档的应用程序

毫无疑问，Internet 已经成为许多公司降低成本、提高效率计划的一部分，另外，Internet 也是销售产品、查询信息的新方法，并能完成许多几年前人们还认为不能完成的许多商务方面的其它任务，这就是本例极为重要的原因。该示例展示了把 Internet 能力置入应用程序的方法，它把两种不同的媒体融合成了一个和谐的整体。

本节展示如何创建基于 HTML 文档的程序框架。虽然用向导生成的框架功能还有限，但它既能显示 Internet 上的 HTML 文档，也能显示不在 Internet 上的 HTML 文档。当我们拥有了可以使用的基本框架后，就可以考虑添加一些绝大多数应用程序都需要的某些增强功能了。

创建基本程序框架

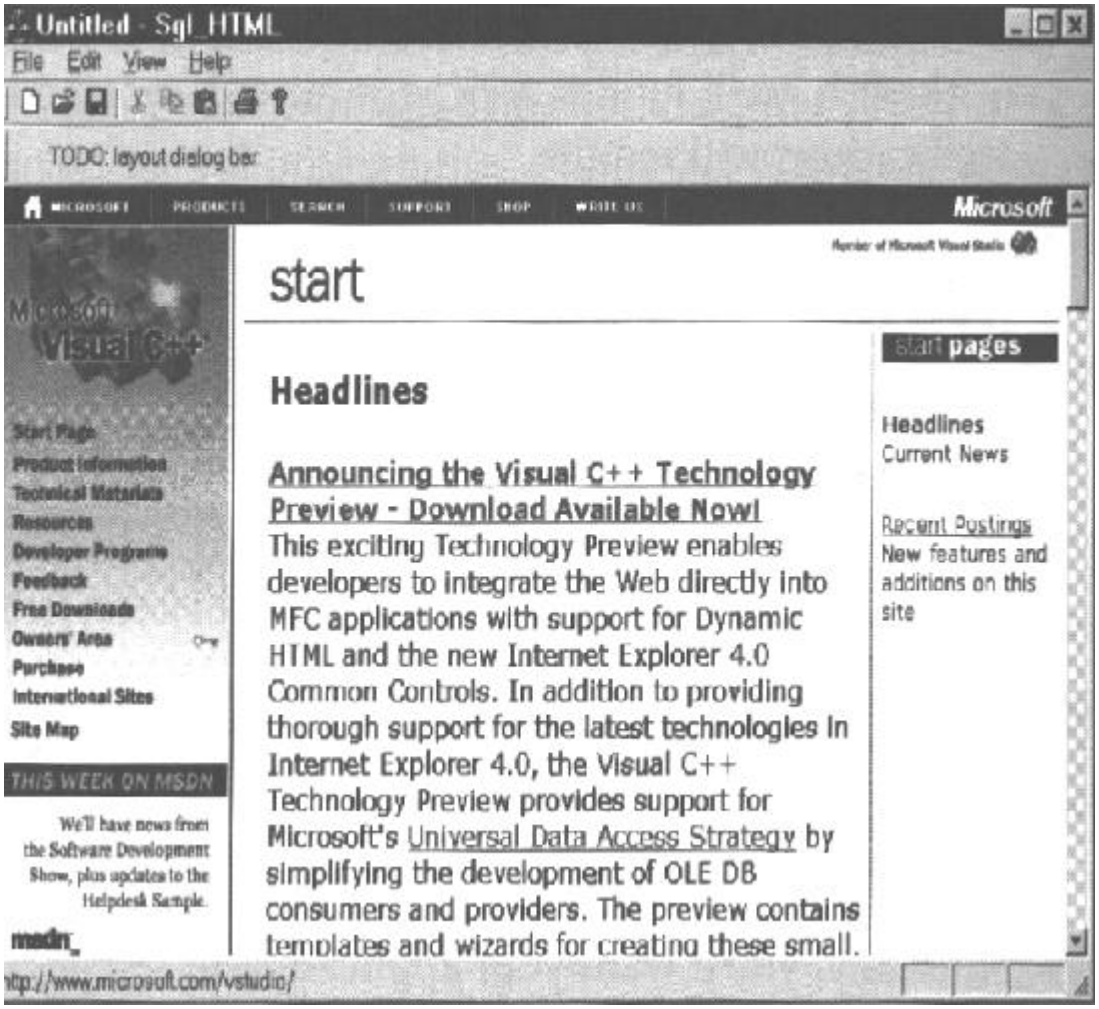
对本例来说，我们将使用本章前面介绍过的单文档应用程序的一种变体。作为本工程的开始，使用 File | New 命令显示 New 对话框，选择 MFC AppWizard(EXE)工程类型，在 Project Name 域中键入名称，示例应用程序使用的应用程序名为 Sg1_HTML，但你可以选择自己喜爱的任何名称。单击 OK 按钮，系统显示 MFC AppWizard - Step 1 对话框，选择 Single Document 选项，然后单击 Next 按钮两次，你会看到 MFC AppWizard - Step 3 对话框。

正常情况下，你会希望你的应用程序既用做 OLE 服务器又用做 OLE 容器。

但针对当前这种应用程序来说，绝大多数用户并不需要 OLE 服务器特性，原因在于只是使用这样的应用程序显示 Web 页面，而无须建立新的数据。正是出于这样的考虑，我们选择 Container 选项并选中 Active Document Container 复选框。回忆一下 Active Document Container 复选框的功能，就会知道它允许你显示像 Microsoft Word 这样的应用程序生成的 ActiveX 文档。再选中 Automation 和 ActiveX Controls 复选框，单击 Next，系统显示 MFC AppWizard - Step 4 对话框。

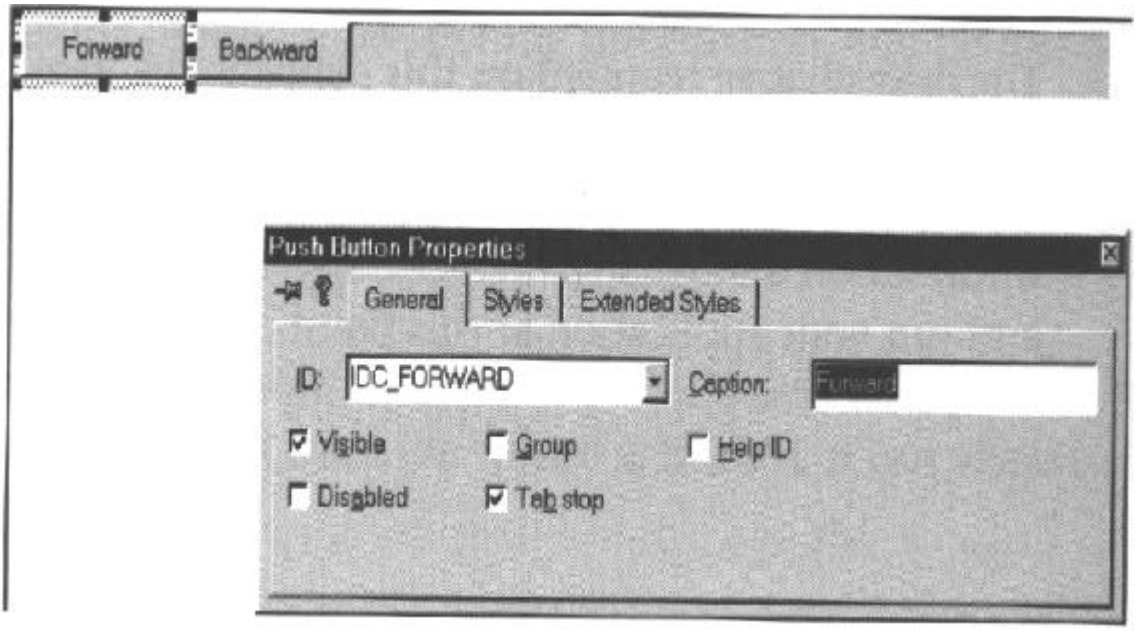
在这个对话框中你至少要看一看其中的几个选项：第一个选项是 Windows Sockets，当应用程序要访问 Internet 或内联网时，应用程序需要 Windows Socket 支持。如果应用程序还要支持电子邮件，那么也要选中 MAPI 选项。最后，由于你的应用程序或许还需要相当复杂的工具条来向用户提供全功能的导航能力，那么也要选中 Internet Explorer ReBars 选项。我还把最近使用文件列表的文件数设置成了 16。单击 Next 两次，系统显示 MFC AppWizard - Step 6 对话框。

这个对话框中唯一需要修改的选项是 Base Class，必须把这个选项设置为 CHtmlView，以便应用程序得到我们所期待的 Internet 功能。现在我们已经配置好了这个示例程序，单击 Finish 按钮，系统显示 New Project Information 对话框，然后单击 OK 生成框架应用。此时编译和运行应用程序时，你会看到如下图所示的屏幕显示。



增强示例程序功能

如果你只是打算显示你的 Web 站点的主页，那么使用上一节创建的应用程序框架就能完成这个任务。然而，在绝大多数情况下，还需要给应用程序添加一些基本的导航能力。只有单一的链接或许并不能让用户满意，因此，该应用程序至少要增加的功能是“向前”与“向后”按钮。现在就让我们看看增加这两个按钮的方法吧。



首先需要完成的任务是打开 IDR_MAINFRAME **对话框，没错，**Internet Explorer ReBar **实际上是个对话框，而不是标准的工具条。该对话框的类是** CDialogBar **而不是过去我们常用的标准** CDialog **类。与其它任何对话框相同，你可以向这个对话框上添加标准的** Windows **控件，我们在这个对话框上添加了两个按钮，如下图所示。请注意，这两个按钮的标题为** Forward **和** Backward**，其** ID **分别为** IDC_FORWARD **和** IDC_BACKWARD**。**

现在我们在 ReBar **对话框上放置了两个按钮，但还没有定义单击它们时做些什么事。实际上，如果现在就编译和运行这个应用程序，你会看到这两个按钮以灰色显示，并且什么事也做不了，其原因在于，我们告诉了** Visual C++**在哪里绘制这两个按钮，但并没有告诉** Visual C++**用它们来干什么。让这两个按钮发挥作用需要下述三个步骤：**

1. **定义按钮与消息处理程序之间的连接。**

2. **声明消息处理程序函数。**

3. **编写消息处理程序代码。**

让我们先完成第一步，定义按钮与消息处理程序之间的连接。由于我们希望这两个按钮影响 HTML **页面的内容，我们需要在** CSgl_HTMLView **类中使用** ON_COMMAND **宏建立连接。程序列表** 2.5 **以粗体形式显示了需要添加到** CSgl_HTMLView.CPP **文件中** MESSAGE_MAP **节的代码。请注意，** ON_COMMAND **宏提供了这两个按钮（标识为** IDC_FORWARD **和** IDC_BACKWARD**）与其对应的消息处理函数之间的连接。**

程序列表 2.5

```
BEGIN_MESSAGE_MAP(CSgl_HTMLView,CHtmlView)
    //{{AFX_MSG_MAP(CSgl_HTMLView)
        // NOTE - the ClassWizard will add and remove    mapping macros here.
        // DO NOT EDIT what you see in these blocks of generated code!
    ON_WM_DESTROY()
    ON_WM_SETFOCUS()
    ON_WM_SIZE()
    ON_COMMAND(ID_OLE_INSERT_NEW,OnInsertObject)
    ON_COMMAND(ID_CANCEL_EDIT_CNTR,OnCancelEditCntr)
    // }}AFX_MSG_MAP
    // Standard printing commands
    ON_COMMAND(ID_FILE_PRINT, CHtmlView::OnFilePrint)
```

```
    ON_COMMAND(ID_FILE_PRINT_DIRECT CHtmlView::OnFilePrint)

ON_COMMAND(ID_FILE_PRINT_PREVIEW,CHtmlView::OnFilePrintPreview)

    //Add two macros for our buttons
    ON_COMMAND(IDC_FORWARD, CSgl_HTMLView::OnForward)
    ON_COMMAND(IDC_BACKWARD, CSgl_HTMLView::OnBackward)
    END_MESSAGE_MAP()
```

当我们定义了这两个按钮与其对应的消息处理程序之间的连接后，需要在 CSgl_HTMLView.H **文件中声明相应的函数。程序列表** 2.6 **显示了声明这些函数的方法。需要保证的一点是，函数要声明成希望创建的消息处理程序。在这个声明中有几点需要注意：首先，把函数声明成** public **方式，这样在该类的外部也可以访问这些函数；其次，把函数声明成虚函数类型，以便在必要时重载它们。**

程序列表 2.6

```
// Message Handlers
public:
    // Add some movement handlers.
    virtual    void    OnForward();
    virtual    void    OnBackward();
```

这两个消息处理函数相当简单，程序列表 2.7 显示了需要添加到 CSgl_HTMLView.CPP 文件尾部的程序代码（在消息处理程序节）。这两个消息处理程序所做的事情就是直接调用 CHtmlView 类的成员函数 GoForward()和 GoBack()。

程序列表 2.7

```
//////////////////////////////////////////////////////////////////////////////////////////////////
// CSgl_HTMLView message handlers
void    CSgl_HTMLView::OnForward()
{
        GoForward();
}

void    CSgl_HTMLView::OnBackward()
{
        GoBack();
}
```

如果现在编译和运行这个应用程序，就会发现 ReBar 上已经添加了两个按钮。单击 Backward 移动到最近浏览过的前一个 Web 页面，单击 Forward 按钮就会看到刚刚退出的那个页面。这两个按钮的工作方式与它们在 Web 浏览器中的工作方式相同。当然，这还仅仅是个开始，你可以在该对话框工具条上添加

任意个数的定制按钮或其它控件。

第3章　理解Visual C++的资源

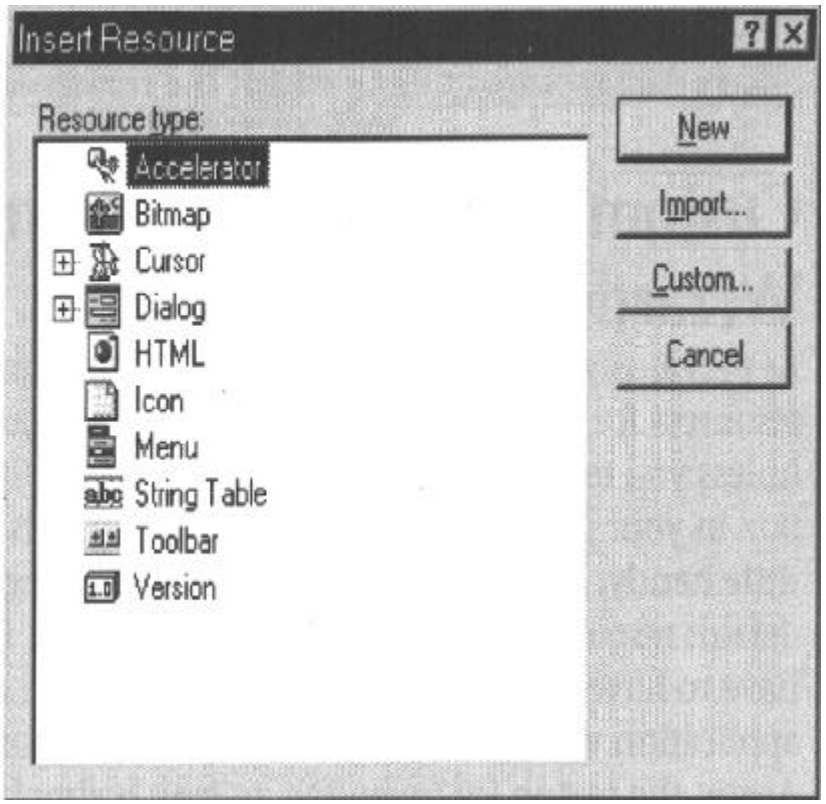
资源，是指各种各样可供利用的财富，它们可能埋藏在地表下面，也可能就在你的后院中。就物理意义而言，从森林中的树木到地层中开采出来的煤炭，这一切都可以称为资源。虽然不能从Visual C++中获得物理意义上的贵重珍宝，但可以挖掘到建立程序时所需的资源。

和任何一种矿产资源一样，Visual C++中的资源提供了一种原材料，将其与程序元素结合起来就可以创建一个成品。在这里，资源与代码结合，创建应用程序的一部分，如菜单或工具条。事实上，在程序中看到的大多数元素也都来自某类资源。当然，Visual C++资源不仅仅是原材料，但原材料的作用是非常大的。我们在下面的章节中讨论资源的用途，牢记住这一点的话，就会得到很大的帮助。

那么，可以从Visual C++的资源矿产中挖掘到什么呢？这些资源的外形和大小各不相同，包括加速键、位图、光标、对话框、图标、菜单、串表、工具条和版本信息。随着课程的进展，我们最后将使用十种资源中的九种（将在第15章讨论HTML资源）。你会看到，某些资源存在子类，每种资源都有特定的用途。在某些情况下，用向导设计程序时，Visual C++自动创建资源，如About对话框。所创建的其它资源是设计另外一些程序的一部分，如字符串。最后，

你会人工设计一些资源，如对话框。

注 Visual C++支持十种资源：加速键、位图、光标、对话框、HTML、图标、菜单、串表、工具条和版本信息。



注释 我们要在本章中使用第二章介绍过的 Sng1_Doc 示例。如果想把这两个练习分开，只需创建 Sng1_Doc 工程文件夹的一个拷贝。为了便于本章使用，请将所有的 Sng1_Doc 工程文件复制到 Resource（资源）工程文件夹。

幸运的是，你可以随时创建任何所需要的资源。只需显示 ResourceView（资源视图），右击 Resources（资源）文件夹，从上下文菜单中选择 Insert（插入）。

你会看到一个 Insert Resource（插入资源）对话框，如上图所示。请注意，它显示出我们已提到过的所有十种资源。还应该看到，有三种创建资源的方式，即使用对话框上的 New（新建），Import（导入）或 Custom（自定义）按钮。

可以在当前程序中使用来自其它工程的资源。例如，可能想在编写的每一个程序中都使用相同的公司徽标。只需右击 Resources（资源）文件夹，从上下文菜单选择 Import（导入）而不是 Insert（插入）。你会看到 Open-type（打开类型）对话框。选择包含了想要导入的资源的文件。正是由于这个原因，你会想到把你的大多数资源与程序分离。可惜的是，这种方法只对图标、光标、声波文件以及 Visual Basic 窗体文件有用。

技巧 有几种使用来自其它工程的其它类型资源的方式，如 About（关于）对话框。只需简单地为想要共享的资源创建一个分离的 RC 文件就行了。这在遇到像 About 对话框这样的资源时很有用。你只要将 RC 文件添加到工程中，剩下的由 Visual C++来处理。另一种共享资源的方法是，将其复制到剪贴板，再在创建了正确类型的空白资源后，将其粘贴到工程中。

自定义资源就是自己设计的资源。实际上，它并不适合 Visual C++所提供的预定义类型。创建一个自定义资源项目很容易：只要单击 Insert Resource（插入资源）对话框上的 Custom（自定义）按钮。你会看到包含单个空白的 New Custom Resource（新建自定义资源）对话框。简单地输入自定义资源名。这时，Visual C++会产生一个新的文件夹，它含有你提供的自定义资源名和该文件夹内的一个新资源。还要提供实现这个资源所需的二进制数据。

3.1 定制使用应用程序向导生成的应用程序所用的资源

如前所述，在设计应用程序的工作区时，Visual C++会自动创建某些类型的资源。例如，除非另外告知 MFC AppWizard，否则它会在程序中包括 About（关于）对话框。这样做的原因是简化应用程序的设计，为开发人员提供一些便利条件，还可以告诉用户是谁设计了他们正在使用的程序。一些缺省资源很有趣，而不是显得很功利。即使你确实不需要特殊的程序图标，还是没有任何办法告诉应用程序向导，你并不想要程序的自定义图标。另外，包括图标的原因也很简单——Windows 需要它在 Explorer 中显示程序。创建的每一个工程都应该有版本信息。这一次这个信息对你是有益处的，因为它帮助你掌握用户拥有哪一种版本的产品。

注 Visual C++ 至少要提供下述缺省资源：自定义应用程序图标、About（关于）对话框和版本信息。

如你所见，自动创建这些缺省资源的原因是很明显的。事实上，这三种资源代表了对每个工程你都应该考虑定制的部件。尽管你应该考虑自己的标准定制技术，但是本节仍会给出一些关于处理缺省资源的提示和技巧。

应用程序图标

用 Visual C++创建的每一个 MFC 应用程序都有一个缺省的应用程序图标。事实上，图标的名称总是相同的：IDR_MAINFRAME。你会发现，这个图标不

仅定义程序图标在 Explorer（或任何其它显示程序图标的程序）中的外观，还同样影响程序的内部表现。例如，About（关于）对话框显示这个图标作为向用户介绍应用程序信息的一部分。

你所创建的所有 MFC 应用程序开始时都有一个相同的图标，如图 3.1 所示。

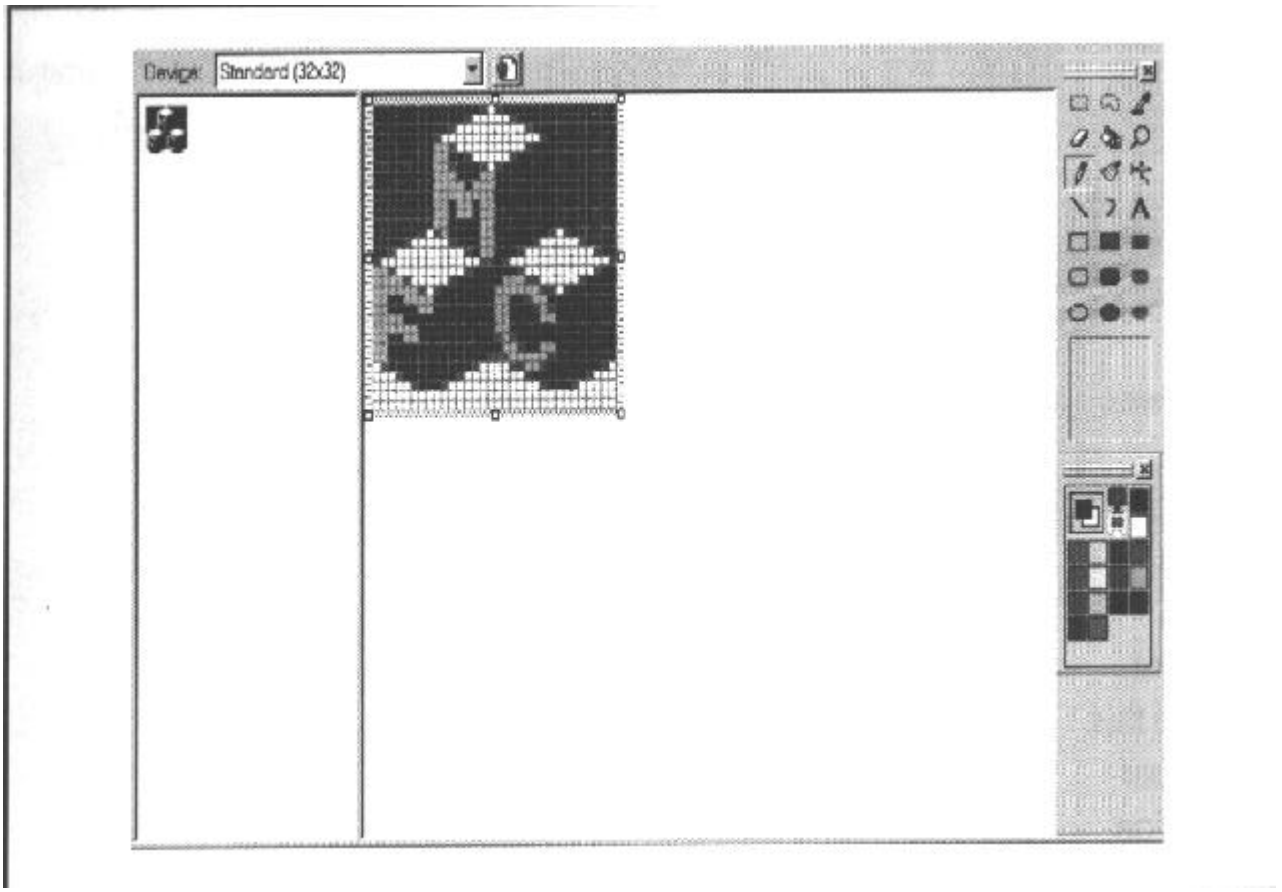


图 3.1　MFC AppWizard 提供了这个可以使用的缺省图标，但你确实应该定制一个自己的图标

注　虽然可以在应用程序中添加许多需要的图标，但缺省的自定义应用程序图标是 IDR_MAINFRAME。

如果你并不在乎人们在安装你的应用程序时看到的图标是什么，那么他们就会看到这个 MFC 徽记。就我个人来说，希望徽记看上去更有趣一些，你也应该如此。毕竟，每次程序用户在 Explorer 中寻找或从 Start（开始）菜单中选择时，都会看到这个图标。

技巧 一定要为应用程序同时修改 32×32 图标（如图 3.1 所示）和 16×16 图标。32x32 图标是在 Explorer 中看到的图标。16×16 图标显示在程序的控制菜单中和 Windows 任务条上。Device 下拉列表框（如图 3.1 所示）是选择 32×32 或 16×16 图标的位置。

决定用应用程序创建文档时，出现第二个缺省图标，就像我们在第二章的 Sng1_Doc 实例中所做的一样。在这种情况下，会看到已创建的每一个文档类型的 IDR。在我们的实例中，它是 IDR_SNGL_DTYPE。它们会有相同的起始图标；图 3.2 显示了 MFC AppWizard 提供的缺省图标。与应用程序图标不同，如果应用程序支持多种文档类型的话，它几乎总是要强制你定制文档的图标。

除了绘图功能外，如果想创建有效的图标，还要了解可供随意使用的工具。所需要的全部工具都显示在窗口的右侧。它们包括一套标准的绘图工具（Graphics 工具条）和一个颜色图表(Colors 工具条），这些工具条可以象我们讨论过的任何其它工具条那样隐藏或显示它们。

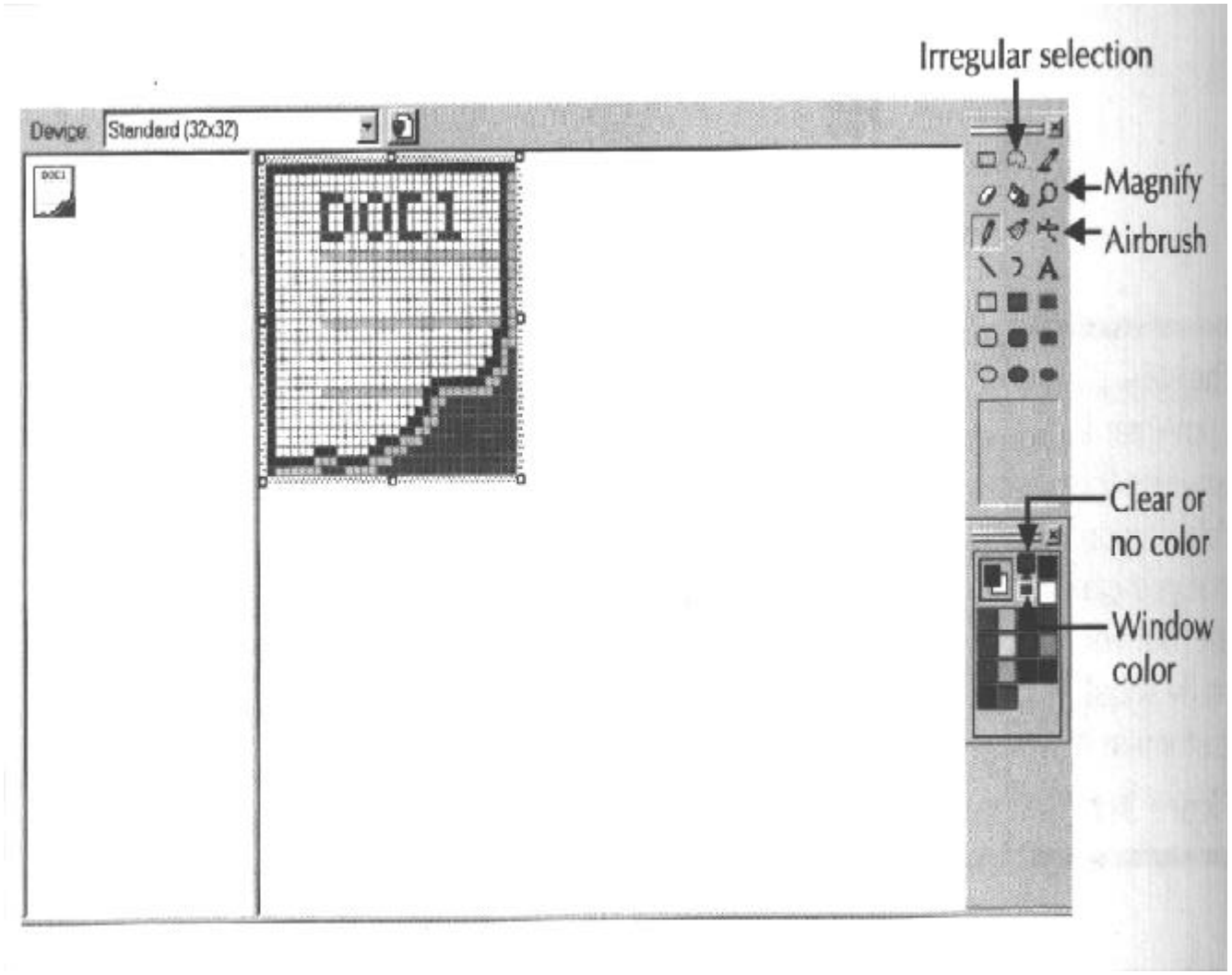


图 3.2　**如果应用程序需要，**MFC AppWizard **还会提供一个像这样的文档图标**

如图 3.2 **所示的** Graphics **工具条上的工具你不应该都不熟悉。你已经熟悉了标准的选择工具、画线和各种椭圆及矩形的工具。但是，**Visual C++**还包括喷雾器工具和不规则选择工具，它们使绘画变得更加容易。你还会发现，**Magnify**（放大）工具比一些绘图程序中的相同工具更为出众——它可以将图画放大至**

正常大小的八倍。

你还需要了解 Colors 工具条上的两种特殊颜色，由于它们使用监视器符号代替了彩色方框，所以可以很容易地找到它们。

注 Colors 工具条中的两个监视器包含了当前窗口颜色和透明颜色，后者可以把图标下面的东西显示出来。

上面的监视器可以创建一个透明区。换言之，你会看到桌面上或放置程序图标的其它地方的该区下面所出现的东西。下面的显示器允许你创建一个区，它使用和用户窗口前景颜色相同的颜色。换言之，改变窗口颜色时，图标中该区的颜色也随之改变。你会看到当前的前景和背景颜色出现在两个监视器的左侧。前景颜色出现在上面的矩形中，背景颜色出现在下面的矩形中。

让我们来看一个示例，看看可以用这两个图标做些什么。图 3.3 显示了为我的程序版本而绘制的示例图标。和你看到的其它图标相比，也许它们的艺术性不是很高——你们当中一定有艺术家——但比你从 Visual C++得到的缺省图标要好些。很明显，可以用任何方式定制图标。请试着使用各种颜色。一定要试验所提到的两种自定义颜色，因为在创建图标时它们特别重要（很多程序员因为不知道如何有效使用这两种特殊的颜色，所以创建的图标看上去很奇怪，它们确实不能与桌面上其它图标协调一致）。

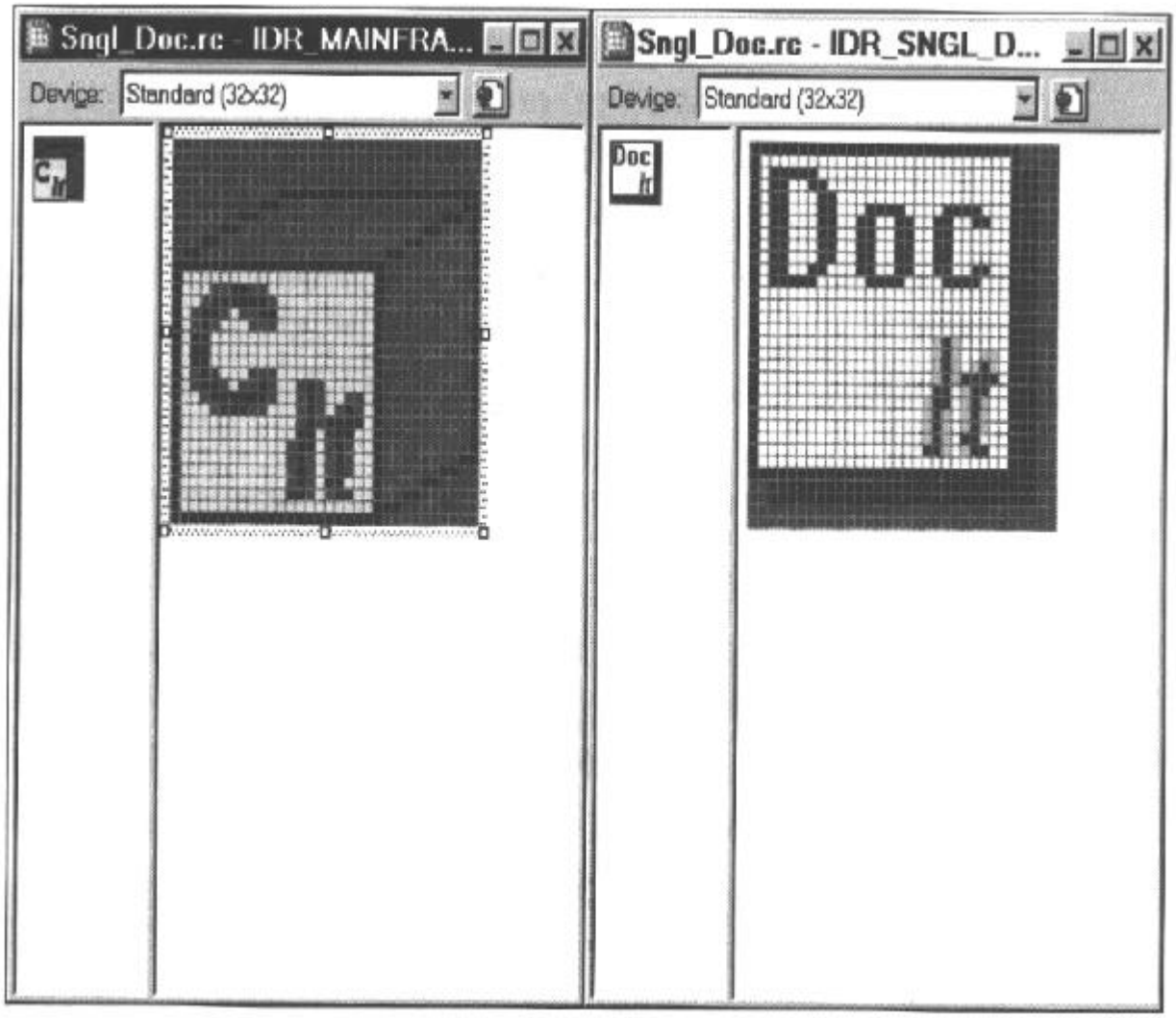


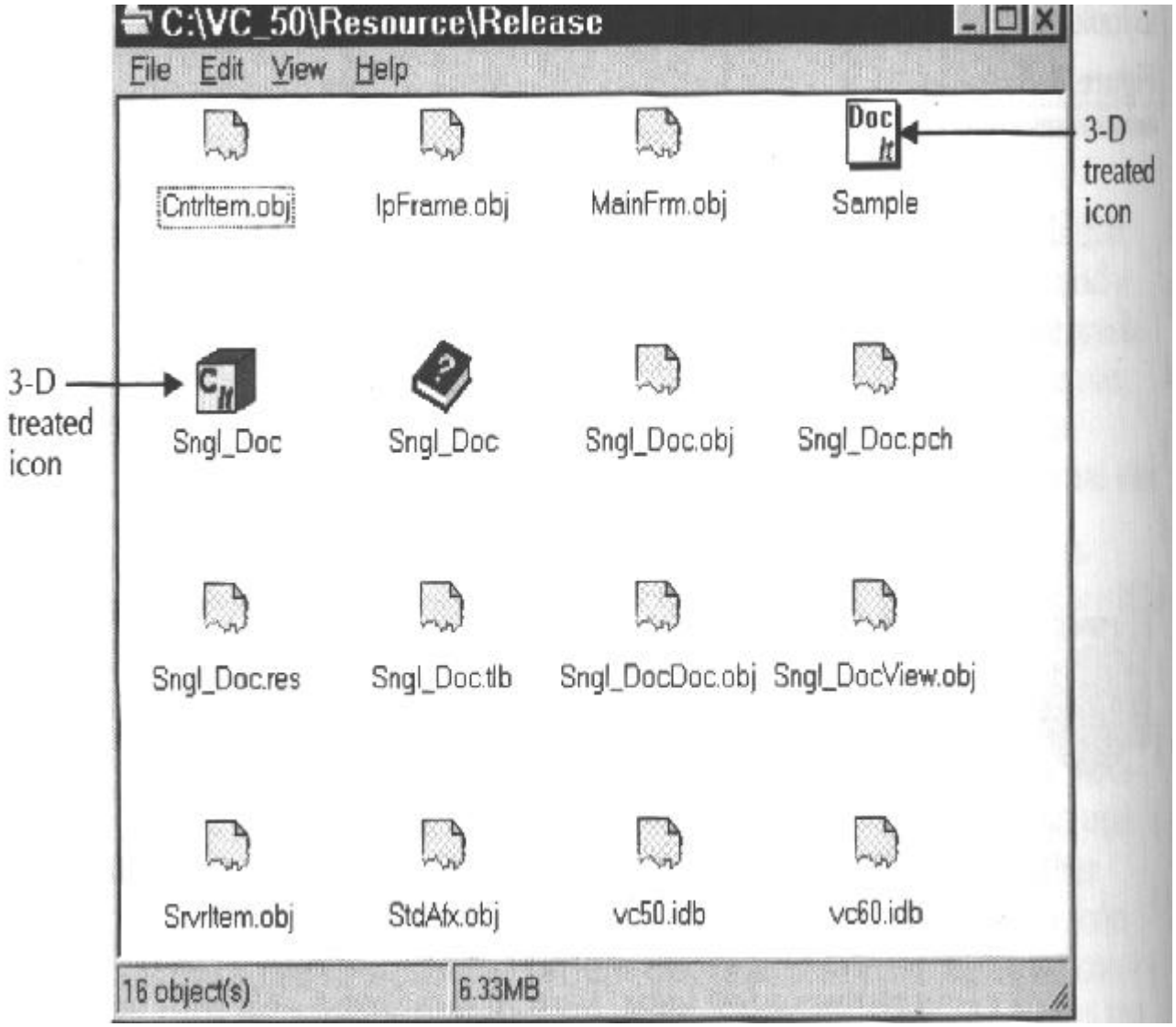
图 3.3　向程序中添加特殊的图标一点也不困难；怎么想的就怎么画

Web 链接　如果创建图标时需要一些美妙的灵感，Internet 上众多的站点正合乎你的需求。一个较好的站点是 http://personal.solutions.net/hillel/ico.htm。它包含了一些很好的光标、图标示例以及创建它们的实用程序。如果这个站点还不能为你提供足够的灵感，可以看一看 http://crab.

看 http://crab. rutgers.edu/icons_new/icons.html。虽然这个站点主要提供GIF 图标，但其 3,000 个示例应该可以带给你一些灵感。这些图标中的绝大多数都涉及流行的主题，像 The Simpsons。幸运的是，图标做了索引，所以你不必一次把全部3,000 个示例都试一遍。你还可以找到数量可观的特定主题站点。例如，http://www.geocities.com/Area51/8604/xfiles.htm 包含一批相当完整的X 文件专用图标（ICO 格式）。

警告 决不要以为你可以在计划出售或转赠他人的程序中使用 Internet 上找到的任何图标（或与此相似的其它图形）。而要假定这些资源有利于引发灵感，除非你得到了原始编写者的书面许可，不可把这些资源用于其它用途。在 Internet 提供的开放环境中，侵权现象是严重的，也比较容易发生。从经验来看，最好的办法是创建你自己的图标，或者从可靠的来源那里得到准许用于商业目的的许可。

请注意，我在两个图标上都使用了透明颜色，并赋予它们三维的外观。再说一句，虽然它们并非那么富于美观，但确实给了用户一种特别的感觉。很明显，编辑器内的透明着色所表现出来的和用户看到（实际上或者看不到）的不同。下面是在 Explorer 内看到的两个图标的外观。



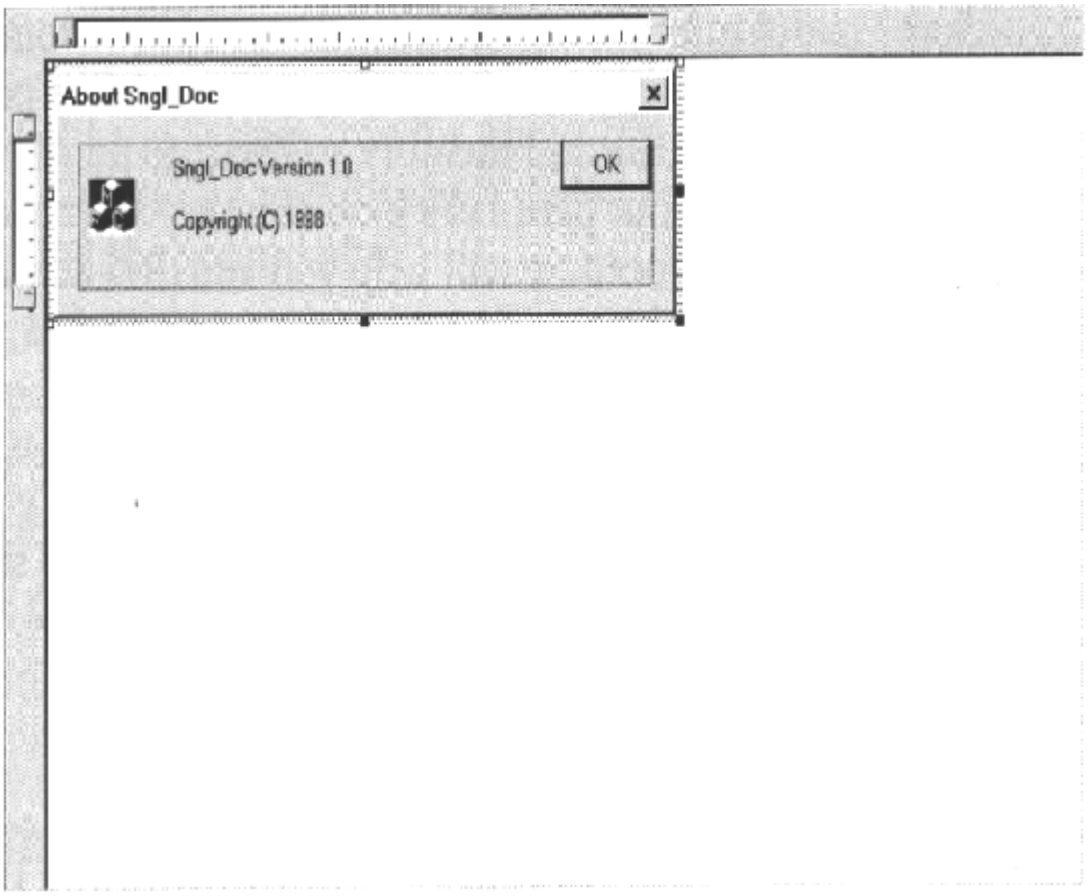
技巧 在修改图标后，一定要再次编译程序，否则它不能在 Explorer 中显示。运行程序也是一个好主意，因为在第一次运行程序时，Visual C++ 会制作注册表项目。最后，一定要在 Explorer 内使用 View | Refresh 命令。否则，你看到的依然是 Explorer 先前存储的老图标。

About 对话框

为应用程序定制 About 对话框并不麻烦。事实上，使用前一节中介绍的信息修改应用程序图标时，就已经开始定制该对话框了。Visual C++自动将你为应用程序创建的图标添加到缺省 About 对话框。下图是我们的示例应用程序使用的缺省 About 对话框。

技巧 因为显示在缺省 About 对话框内的信息总是基于你提供给 MFC AppWizard 的输入，所以要给出尽可能清楚的信息。这样做将减小后续的工作量。这些设置还会影响我们将在本章后面看到的版本信息。

那么，需要在标准的 About 对话框中提供些什么呢？缺省 About 对话框为你提供了一些灵感。基本上，一个 About 对话框应该提供四条信息：公司名称，版权信息，产品名称和版本号。如果提供了这样四个必要的项目，应用程序的 About 对话框就有了它绝对应该拥有的一切。请注意，缺省 About 对话框提供了这四条必要信息中的三条——如果你不想做太多工作的话，也只需要添加上公司名称即可。



当然，你可能想将其它一些必要的信息提供给应用程序中的 About 对话框。例如，让用户知道该去哪里获得技术支持是很重要的。添加电话号码和几个访问指令是个好主意，这也无须消耗太多的空间。一些公司还添加了产品注册序号，作为技术支持信息的一部分，但提供这类信息要从注册表或 INI 文件开始。

某些有用的信息包括了可用于支持应用程序的当前系统资源。例如，Visual C++提供了一种将当前磁盘和内存统计值添加到 About 对话框的方式，而且并不需要你自己多做多少工作。让我们看看如何才能在对话框中添加这种能力。下面的过程帮助你将磁盘和内存统计值添加到所创建的任一 About 对话框（或

者与此相似的任一其它对话框）中。

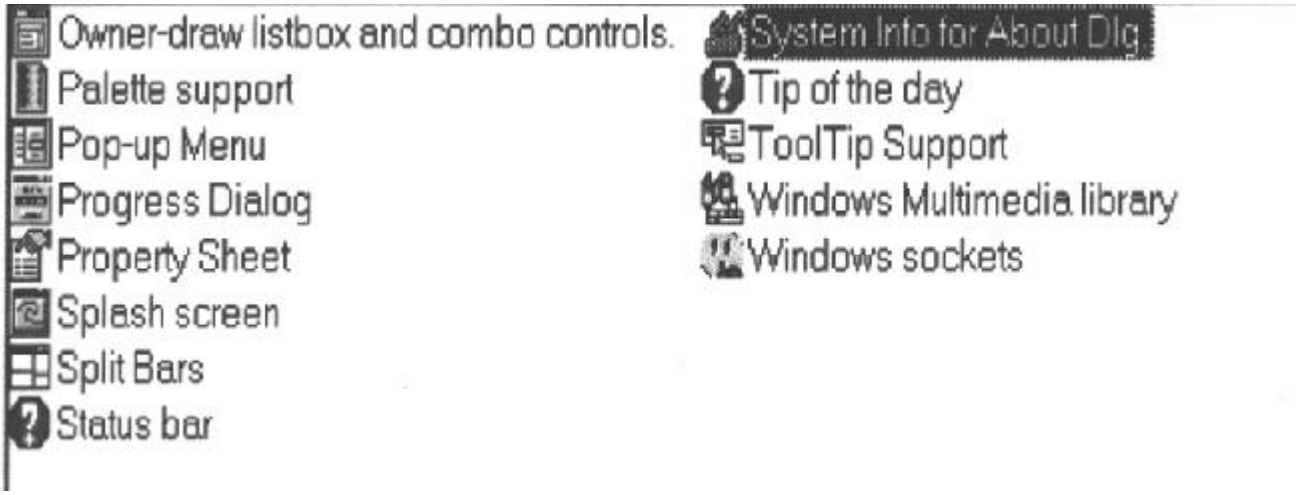
1. 调整 About 对话框的大小。现在的对话框太小，放不下额外的信息。很明显，你也不想把它做得太大，因为 About 对话框应该足够小，以适应可能存在的最小显示区。在本示例中，将 About 对话框的大小调整为 250*150，这应该足以显示磁盘和内存统计值了，也为版权和其它辅助信息留出了空间。

2. 将两个静态文本框控件添加到 About 对话框中。前一个用于显示内存统计值，后一个用于显示磁盘统计值。

3. 使这两个控件的大小为 150*18。将内存统计值静态文本框控件定位在 40,40。将磁盘统计值静态文本控件定位在 40,65（这些定位数字假定，你没有为了适应更多的公司名称或产品信息而改变缺省静态文本框控件的大小）。

4. 右击内存统计值静态文本框控件，然后从上下文菜单中选择 Properties（属性）。将该控件的 ID 字段改成 IDC_MEMORY。对磁盘统计值静态文本框控件做相同的处理。但是，在这个事件中将 ID 字段改变成 IDC_DISK。不要选中两个控件的 Group（组）框——我们想把它们作为独立的静态文本框控件使用。现在，我们有地方放置信息了，可以添加收集信息的代码了。

5. 使用 Project | Add to Project | Components and Controls 显示 Components and Controls Gallery 对话框。在 Look In 字段中双击 Visual C++ Components，你会看到如下图所示的一系列部件。



6. **加亮** System Info for About Dlg **部件，然后单击** Insert（**插入**）。Visual C++ **会显示一个对话框，询问是否确实要安装该部件。**

7. **单击** OK。Visual C++**会显示另一个对话框，告诉你正在安装的部件会添加信息到** About **对话框中。**

8. **单击** Yes **继续安装。**Visual C++**完成** System Info for Abourt Dlg.**部件的安装。**

9. **单击** Close **关闭** Components and Controls Gallery **对话框。**

此时，好像什么事也没有发生。Visual C++**只是停在那儿，什么也不做。除非你知道到哪里去看，否则永远也找不到** System Info for About Dlg.**部件所添加的代码。打开** ClassView，**然后双击** CAboutDlg::OnInitDialog()**函数。你会在初始化代码中看到一些新的条目，按程序列表** 3.1 **所示进行修改。**

程序列表 3.1

```
BOOL CAboutDlg::OnInitDialog( )
```

```
{
        CDialog::OnInitDialog();        // CG: This was added by System Info Compoment.
// CG: Following block was added by System Info Component.
{
        CString strFreeDiskSpace;
        CString strFreeMemory;
        CString strFmt;

        // Fill available memory
        MEMORYSTATUS    MemStat;
        MemStat.dwLength = sizeof(MEMORYSTATUS);
        GlobalMemoryStatus (&MemStat);
        strFmt.LoadString (CG_IDS_PHYSICAL_MEM);
        strFreeMemory.Format(strFmt, MemStat.dwTotalPhys / 1024L);

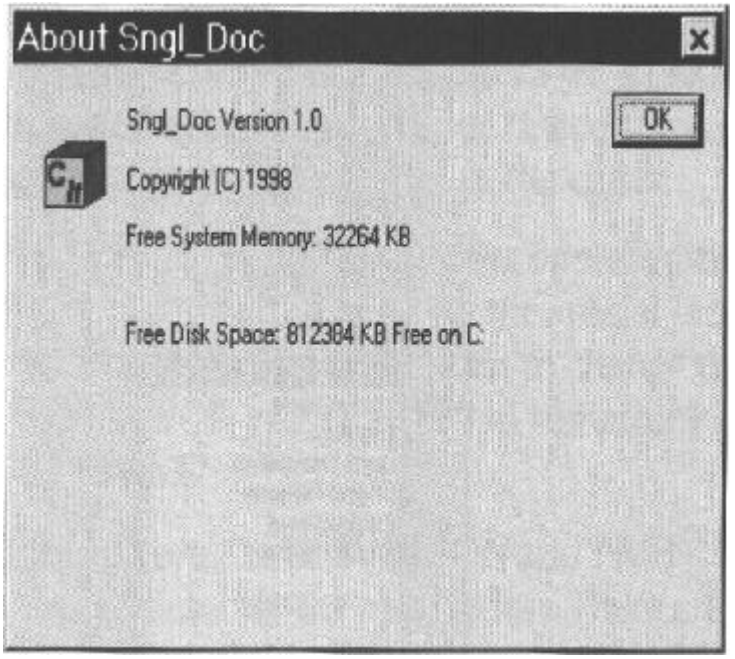
        // Display the amount of free memory.
        strFreeMemory = "Free System Memory:" + strFreeMemory;
        SetDlgItemText (IDC_MEMORY,strFreeMemory);

        // Fill disk free information
        struct_diskfree_t diskfree;
```

```
    int nDrive = _getdrive(); // use current default drive
    if (_getdiskfree(nDrive, &diskfree) == 0)
    {
        strFmt.LoadString(CG_IDS_DISK_SPACE);
        strFreeDiskSpace.Format(strFmt,
            (DWORD)diskfree.avail_clusters *
            (DWORD)diskfree.sectors_per_cluster *
            (DWORD)diskfree.bytes_per_sector /    (DWORD)1024L,
            nDrive-1+_T('A'));
    }
    else
        strFreeDiskSpace.LoadString(CG_IDS_DISK_SPACE_UNAVAIL);
        // Display the amount of free disk space.
        strFreeDiskSpace =  " Free Disk Space: " + strFreeDiskSpace;
        SetDlgItemText(IDC_DISK, strFreeDiskSpace);
    }
    return TRUE;    // CG: This was added by System Info Component.
}
```

请注意，System Info for About Dlg.**部件提供的缺省代码只从缺省驱动器中检索数据。可以很容易地修改这个代码，以检索用户机器上所有驱动器的驱动器空间信息。你得到的内存及磁盘空间字符串缺少像“**Free Disk Space**”这样的**

提示，这一提示说明了 About 对话框提供的是什么信息。可惜的是，没有提示信息将给用户带来更多的疑惑，而不是帮助。一定要添加某种提示信息（像程序列表 3.1 中显示的那样）来告诉用户数据的意义。最后请注意，缺省字符串实际上由更为复杂的统计值组成。可以很容易地将这个额外的信息添加到你的 About 对话框。下面是这个代码产生的最终结果。



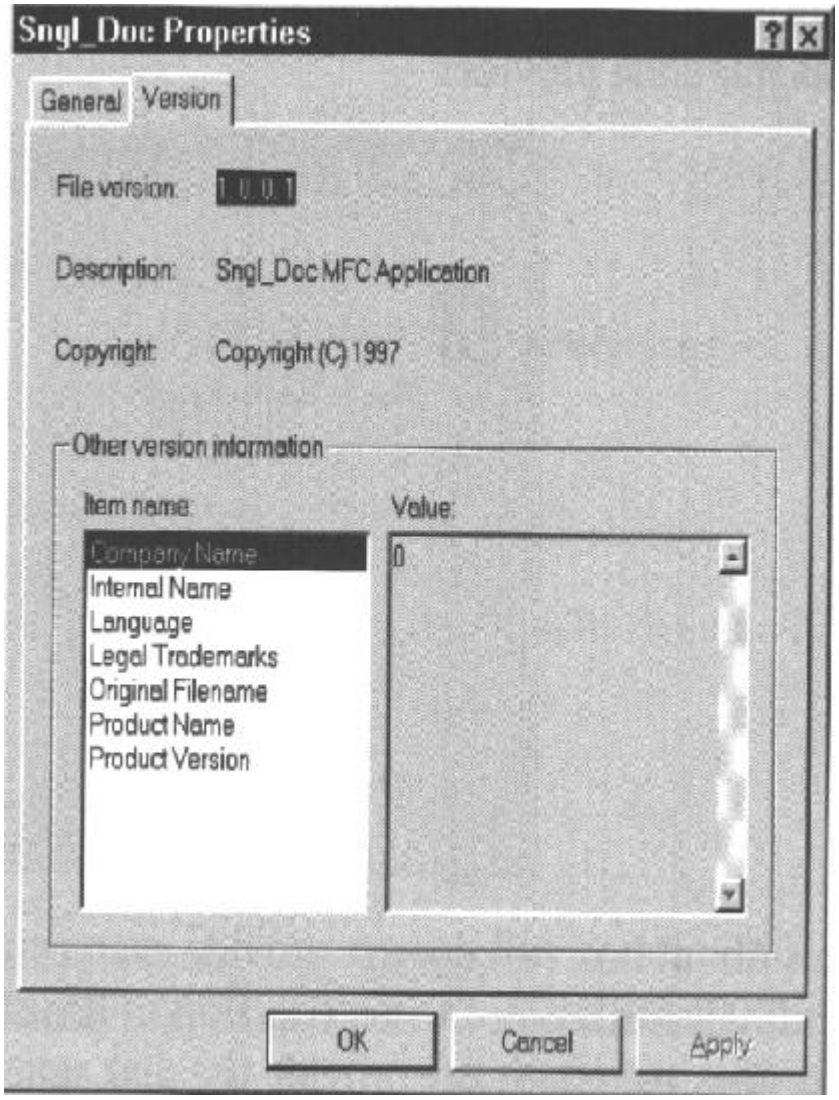
这个 About 对话框并不提供用户所需要的一切，但我们将在本章的下一节中进一步修改它。现在，About 对话框自动提供了磁盘空间和内存的统计值。它还提供了程序名称、版本和版权信息。我们将在下一节看到，你同样可以自动生成这三条信息。

现在再看一看定制问题。谈到定制这个话题，你可以做许许多多的工作。目前，应用程序中的某些 About 对话框本身就是个袖珍型的应用程序，它们或许承担的任务太多了。如果发现有必要向 About 对话框中添加按钮和另外一些

对话框时，就应该考虑将此信息放置到别的地方，比如某个帮助文件中。

版本信息

你或许未认真思索就跳过了 Visual C++自动提供的版本信息资源。以前，除了知道如何找到版本信息的程序员外，该信息对每个人来说都深深地隐藏了起来。问题在于，使用 Windows 95 和 Windows NT 4提供的 Explorer 界面，你不能再回避版本信息了。现在要显示应用程序提供的版本信息，只需右击Explorer中的程序图标，然后从上下文菜单选择 Properties（属性）。选择 Properties（属性）对话框的 Version（版本）选项卡，会看到如下图所示的画面。



如你所见，缺省版本的信息量很少。如果是这样的话，用户甚至会不知道他们在和哪一家公司打交道。因为越来越多的用户开始关注 Properties **（属性）对话框必须提供些什么，所以对程序员来说，在这里放入适当类型的信息就变得越来越重要了。**

技巧 填写应用程序的版本信息并不是单向的。也可以用这个信息填写应用程序的其它区域，这意味着你必须只改变一个地方的信息，使其处于当前状态。在本节中我们看看如何为 About 对话框使用这种技术，当然你也可以在其它地方使用它。

高级技巧

在某些情况下，你或许想修改版本信息资源中的文件专用信息。例如，如果双击 FILEFLAGS（文件标记）条目，会看到 FILEFLAGS（文件标记）对话框包含两个复选框。第一个复选框说明版本信息是调试版的版本信息还是发行版的版本信息。第二个复选框说明该版本信息是否是程序的预发行版。对你的β版程序来说，可以选中 VS_FF_PRERELEASE 复选框。一旦该程序能够正常发行，那么你应该不选中这个复选框。

FILEOS 条目是另一个提供定制机会的地方。假定程序依赖 Windows NT 提供的安全特征，你可能想将 FILEOS 条目从缺省的 VOS_WINDOWS32 改成 VOS_NT 或 VOS_NT_WINDOWS32。

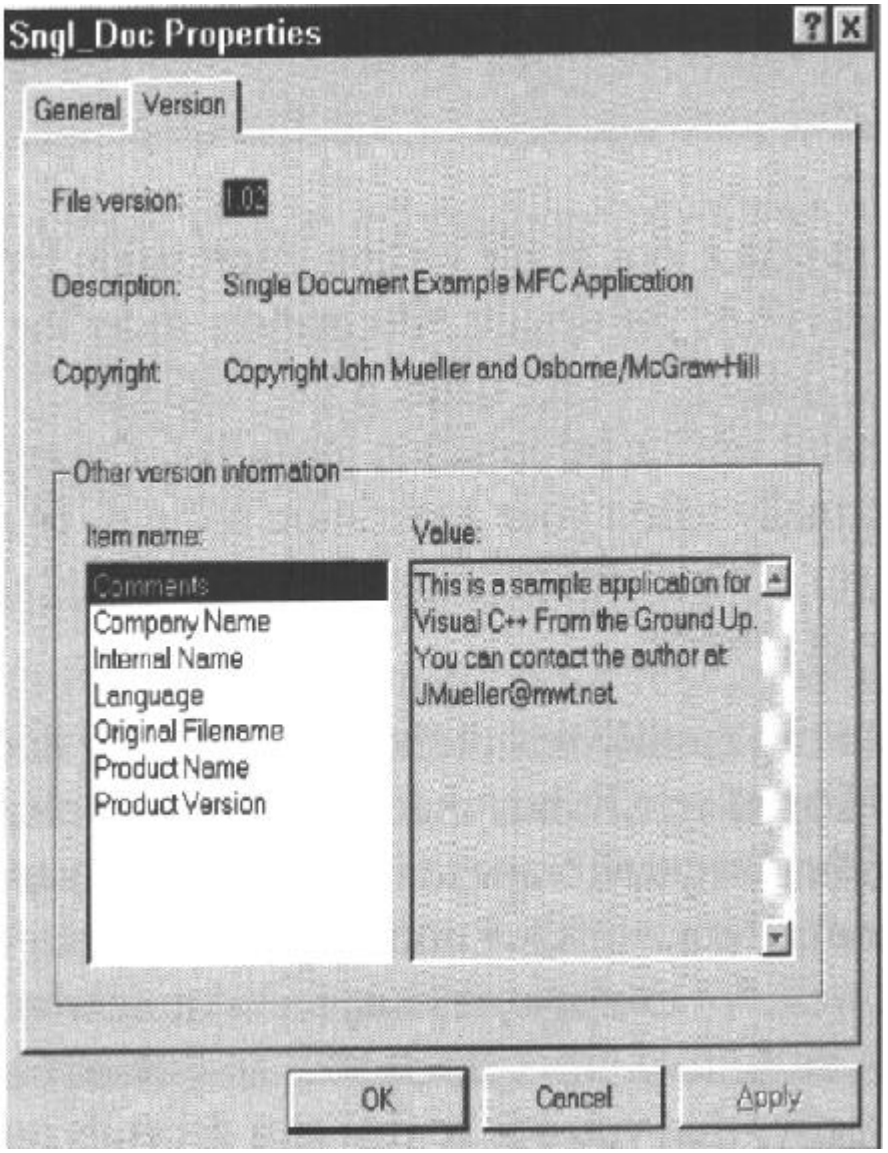
修改应用程序时保持 FILEVERSION（文件版本）和 PRODUCTVERSION（产品版本）域为最新也很重要。使用这些区域有几个不同的方式。最好的方式是，简单地忽略 Visual C++建议的方法，如果这是程序的第一个版本，输入像 1.0 这样的简单数字。

让我们看一看示例应用程序的版本信息。可以在 ResourceView（资源视图）中的 Version（版本）文件夹下找到缺省信息。缺省资源名是 VC_VERSION_INFO。图 3.4 显示了缺省的版本信息。粗实线上的条目通常反映应用程序的设置，并不需要经常修改它们。以 Block Header 开始的粗实线下



图 3.4　要修改缺省版本信息，使其和公司及产品的实际信息一致

可以双击任何一个文本条目来修改它们。Visual C++会打开一个用来更改信息的编辑框。至少，你想更新 CompanyName（公司名称），LegalCopyright（合法版权）和 ProductName（产品名称）字段。我通常还添加一些信息到 Comments（注释）字段。例如，关于某个应用程序或其它可执行程序，知道该与谁联系是很方便的，所以我经常添加我的名字和电子邮件地址。到底向这个区域添加些什么，要视公司政策、合法需要和个人爱好而定。下面是在 Properties（属性）对话框中查看时，我的已修改版本的信息。



既然已经花时间向应用程序中添加了版本信息，那么充分发挥这些信息的作用就显得尤其重要了。About **对话框中相同类型的信息通常还出现在其它需要版本信息的某些地方，所以将两者结合起来是一个好主意。这样就不会出现** About **对话框和版本信息不合拍的情形。程序列表** 3.2 **显示了这样的代码，你需要将其添加到** CAboutDlg::OnInitDialog()**函数中，以将版本信息添加到** About **对**

话框（我们在本章的 About 对话框一节中看到过这个函数，所以这里只说明新的代码——将此代码放在函数的开始位置）。

程序列表 3.2

```
LPTSTR      lpstrFileName;      // Name of our program.
DWORD       dwHandle;           // A placeholder handle.
DWORD       dwSize;             // Size of the version information block.
LPVOID      lpBuffer;           // Buffer to store version information block.
LPVOID      lpData;             // One version information value.
UINT        uiDataSize;         // Size of the version information value.
Cstring     strProduct;         // Product name and version.
Cstring     strCompany;         // Company name and copyright information.

// Initialize our variables and allocate memory.
lpstrFileName = "Sngl_Doc.EXE";
dwHandle = 0;
uiDataSize = 80;
lpData = malloc(uiDataSize);

// Get the version information block size,
// then use it to allocate a storage buffer.
dwSize = GetFileVersionInfoSize(lpstrFileName, &dwHandle);
```

```
lpBuffer = malloc(dwSize);

// Get the version information block.
GetFileVersionInfo(lpstrFileName, 0, dwSize, lpBuffer);

// Use the version information block to obtain the product name.
VerQueryValue(lpBuffer,
      TEXT("\\StringFileInfo\\040904b0\\ProductName"),
      &lpData,
      &uiDataSize);
strProduct = LPTSTR(lpData);
strProduct = strProduct + "\n";

// Use the version information block to obtain the product version.
VerQueryValue(lpBuffer,
      TEXT("\\StringFileInfo\\040904B0\\ProductVersion"),
      &lpData,
      &uiDataSize);
strProduct = strProduct + LPTSTR(lpData);

// Display the product name and version.
SetDlgItemText(IDC_PRODUCT,strProduct);
```

```
// Use the version information block to obtain the company name.
VerQueryValue(lpBuffer,
        TEXT("\\StringFileInfo\\040904B0\\CompanyName"),
        &lpData,
        &uiDataSize);
strCompany = LPTSTR(lpData);
strCompany = strCompany + "\n";

// Use the version information block to obtain the copyright information.
VerQueryValue(lpBuffer,
TEXT("\\StringFileInfo\\040904B0\\LegalCopyright"),
         &lpData,
         &uiDataSize);
strCompany = strCompany + LPTSTR(lpData);

// Display the company name and copyright information.
SetDlgItemText(IDC_COMPANY, strCompany);

// Free the memory we allocated.
free(lpBuffer);
free(lpData);
```

这段代码比你一开始想像的要更容易理解一些。实际上，可以将整个过程

分解成如下四个简单的步骤：

1. 用 GetFileVersionInfoSize()函数获得 VS_VERSION_INFO 结构的大小。

2. 用 VS_VERSION_INFO 结构大小创建一个足以保持该结构的缓冲区，然后用 GetFileVersionInfo()函数获得它。

3. 一旦有了 VS_VERSION_INFO 结构的本地副本，就用 VerQueryValue()函数获得各个字符串。

4. 在 About 对话框中显示字符串。

现在，你已经很好地理解了程序的总流程，让我们看看一些细节。你想知道的第一件事，可能是 VerQueryValue()函数调用中的下面一行字符串：

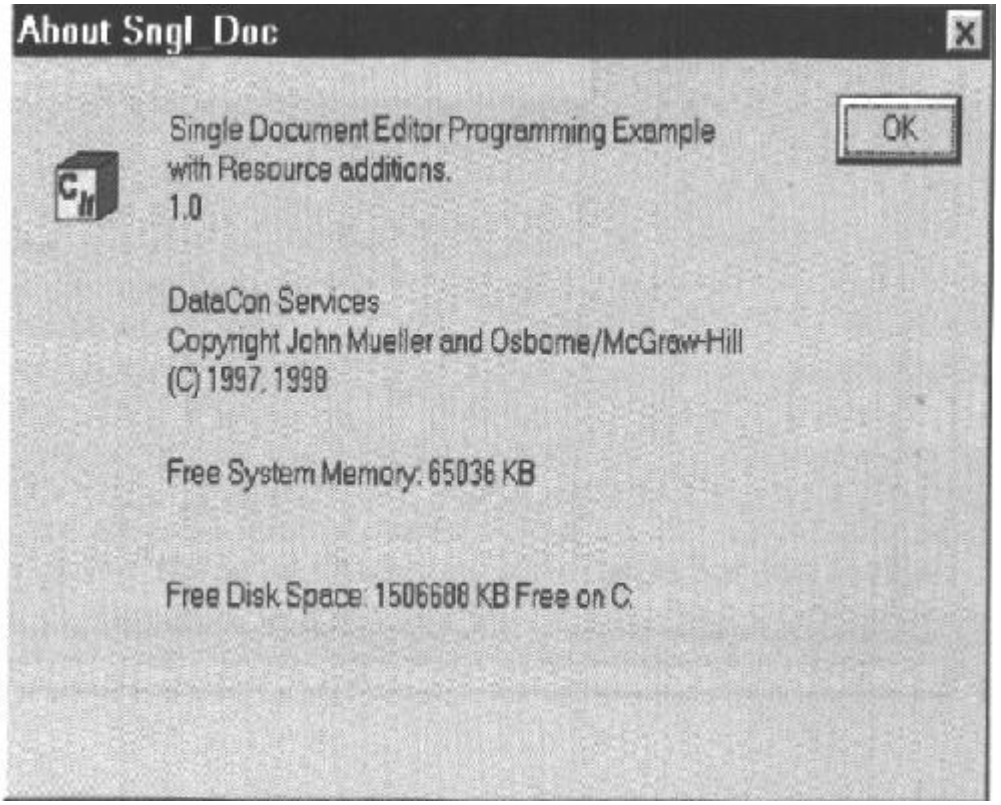
```
TEXT("\\StringFileInfo\\040904b0\\ProductVersion")
```

该字符串的第一部分 StringFileInfo 说明我们要在 VS_VERSION_INFO 结构中引用什么样的结构。在处理图 3.4 中粗实线下的信息时，StringFileInfo 是一个常量。第二参数说明想使用信息的语言版本。请注意，图 3.4 中的 Block Header（块标题）条目说明了我们正使用英语（美国）。在该条目的右侧，会看到一串 8 个数字，刚好与我们字符串的数字相匹配。从这里获得 VerQueryValue()调用的数字。最后，该字符串的最后一部分是版本信息块中字符串的一部分。在样本字符串的事件中，我们正在寻找 ProductVersion 字符串。如果仔细浏览程序列表 3.2 中的示例代码你会注意到，从一个 VerQueryValue()调用到下一个调用，唯一发生改变的，是我们要查找的字符串。

在编译程序之前，还要注意最后一件事。用 Project | Settings（工程|设置）命令显示 Project Settings（工程设置）对话框。选择对话框的 Link（链接）选

项卡。我们要添加一个特殊的库到程序中，这样才能访问版本信息。在Object/Library Modules（对象/库模块）字段中添加一个名称为 VERSION.LIB 的条目，然后单击 OK。

如你所见，作为一段可以反复使用的代码（编写新程序时，即使需要修改的话，也不会改得太多），你得到了一个能自动更新自身的 About 对话框。下面是 About 对话框的外观。请注意，我重新安排了静态文本框控件的位置，还分别赋予了两个缺省控件新的标识 IDC_PRODUCT 和 IDC_COMPANY。



3.2 使用加速键和菜单

菜单和加速键密切地联系在一起。这两种资源类型联手协作，让用户能更容易地完成任务。大家都知道菜单是什么——它是分层命令结构的物理表示。加速键提供该结构的快捷方式，以提高用户操作的速度。例如，要创建一个新文件，通常用 File | New（文件|新建）命令或 CTRL-N 加速键，两种方法可以产生相同的结果。

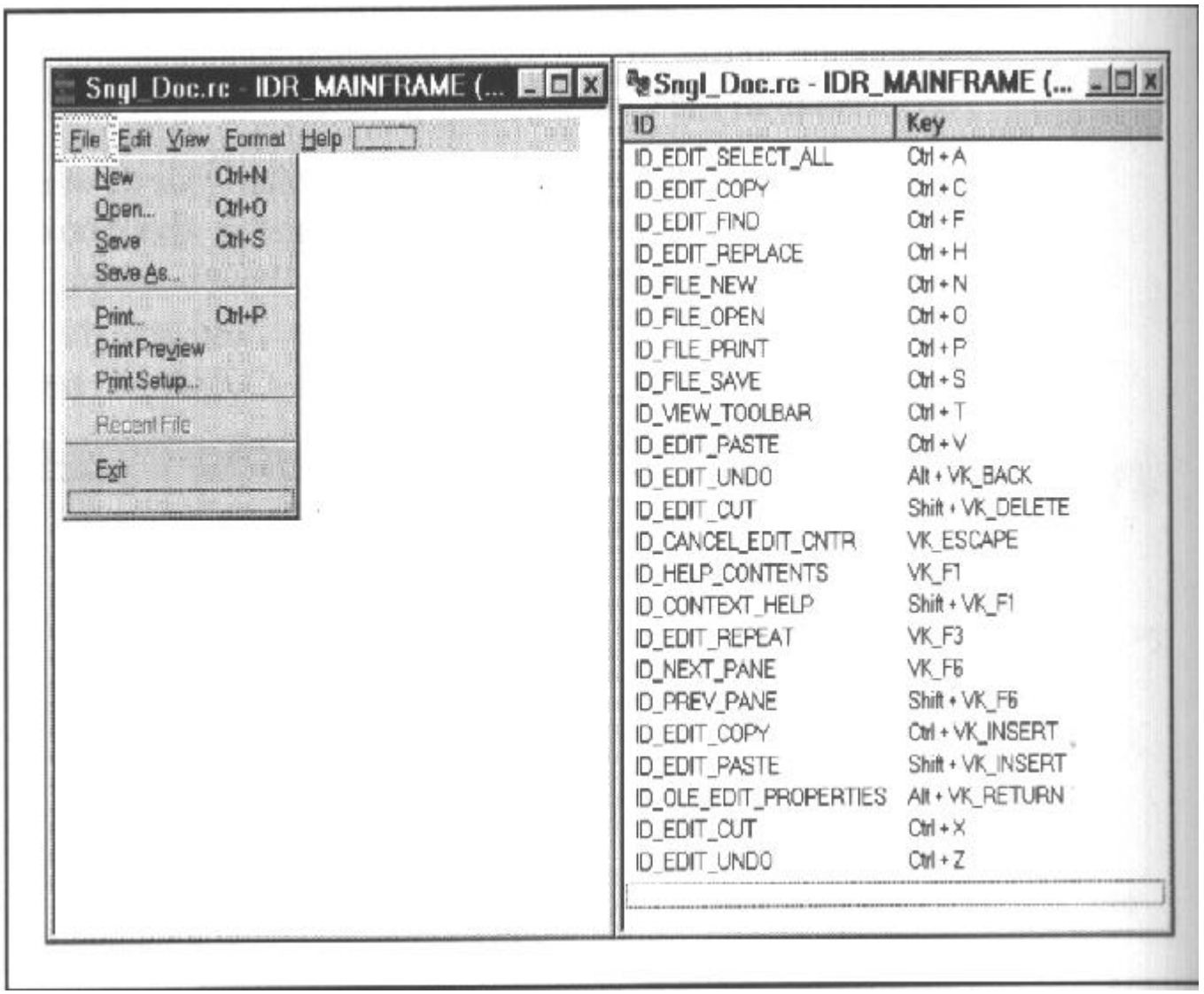
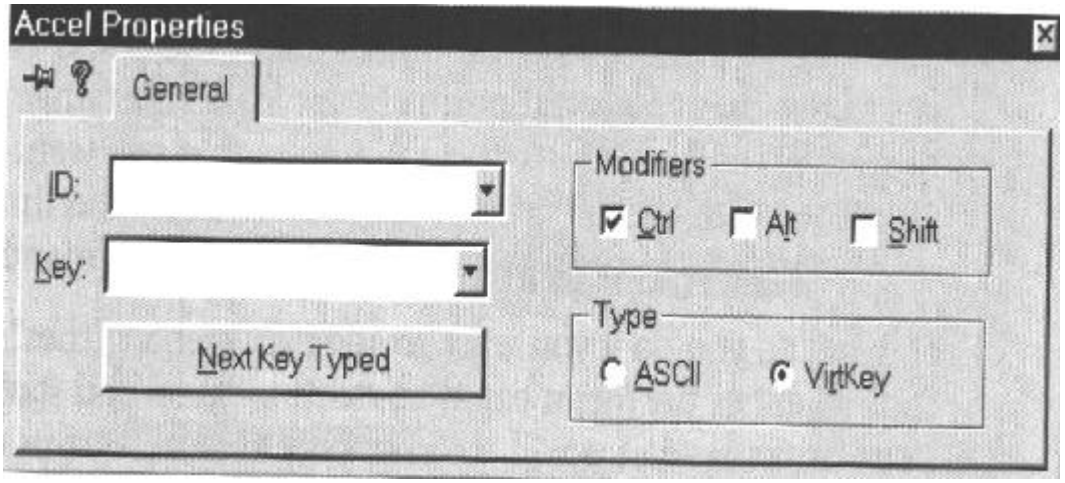


图 3.5　缺省菜单提供了你所期待的标准功能。加速键资源通过你给它赋于的名称与相应的菜单联系在一起

Visual C++**将菜单和加速键作为两种不同的资源存储。图** 3.5 **显示了示例应用程序的主菜单和相应的加速键。注意有一点很有意思，两种资源使用相同的名称** IDR_MAINFRAME**。应该记住这一点，因为该名称是将两种资源（菜单和**

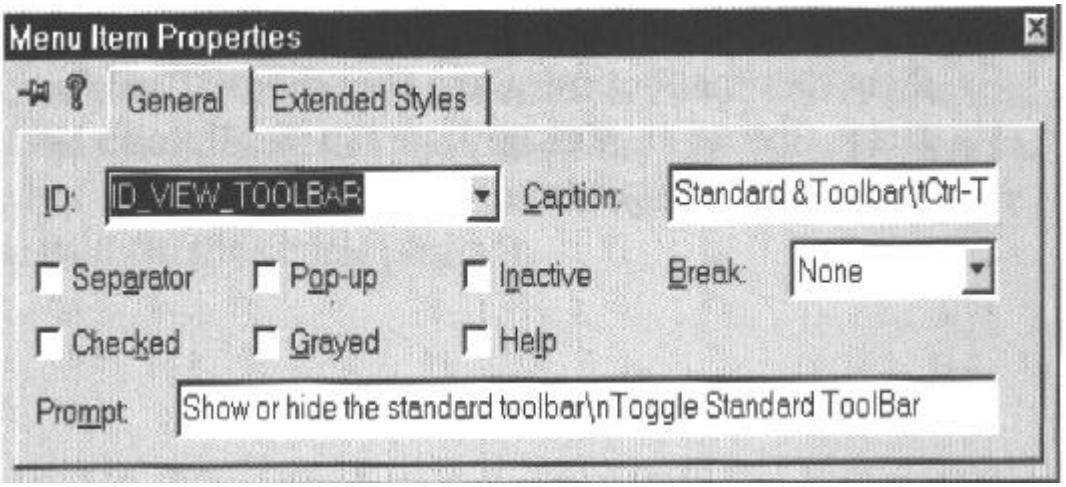
加速键）链接在一起的资源名。



让我们看看菜单和加速键是怎样协作的。将新的条目添加到现有的加速键资源中，这一点很容易。在 Accelerator（加速键）窗口中右击，然后从上下文菜单中选择 New Accelerator（新加速键）。就会看到 Accel Properties（加速键属性）对话框，如下图所示。

此时，需要从下拉列表框中选择一个 ID。Menu ID 通常使用所需菜单层次的组合来得到名称，然后在名称前加上字符 ID。例如，如果想为 View | Toolbar（视图|工具条）命令创建一个加速键（我在实例中就是这样做的），就应在 ID 字段中选择 ID_VIEW_TOOLBAR。也可以在 ASCII 码和虚拟键（VirtKey）之间进行选择。当你准备添加与某个菜单命令相关的按键时，只需单击 Next Key Typed 按钮，你会看到一个比较小的对话框，告诉你按下想要当作加速键使用的组合键。在这个示例程序中，我按下 CTRL-T。你会看到，CTRL 复选框中有一个复选标记，Key 域中有一个"T"，单击 Close（关闭）方框结束添加加速键操作。

如果现在就编译并运行程序，刚刚添加的加速键即能正常工作。事实上，你可能正想这样。但是，用户可能想不到，快速执行一个菜单命令可以用加速键。要向菜单中添加加速键，需要修改当前的菜单。

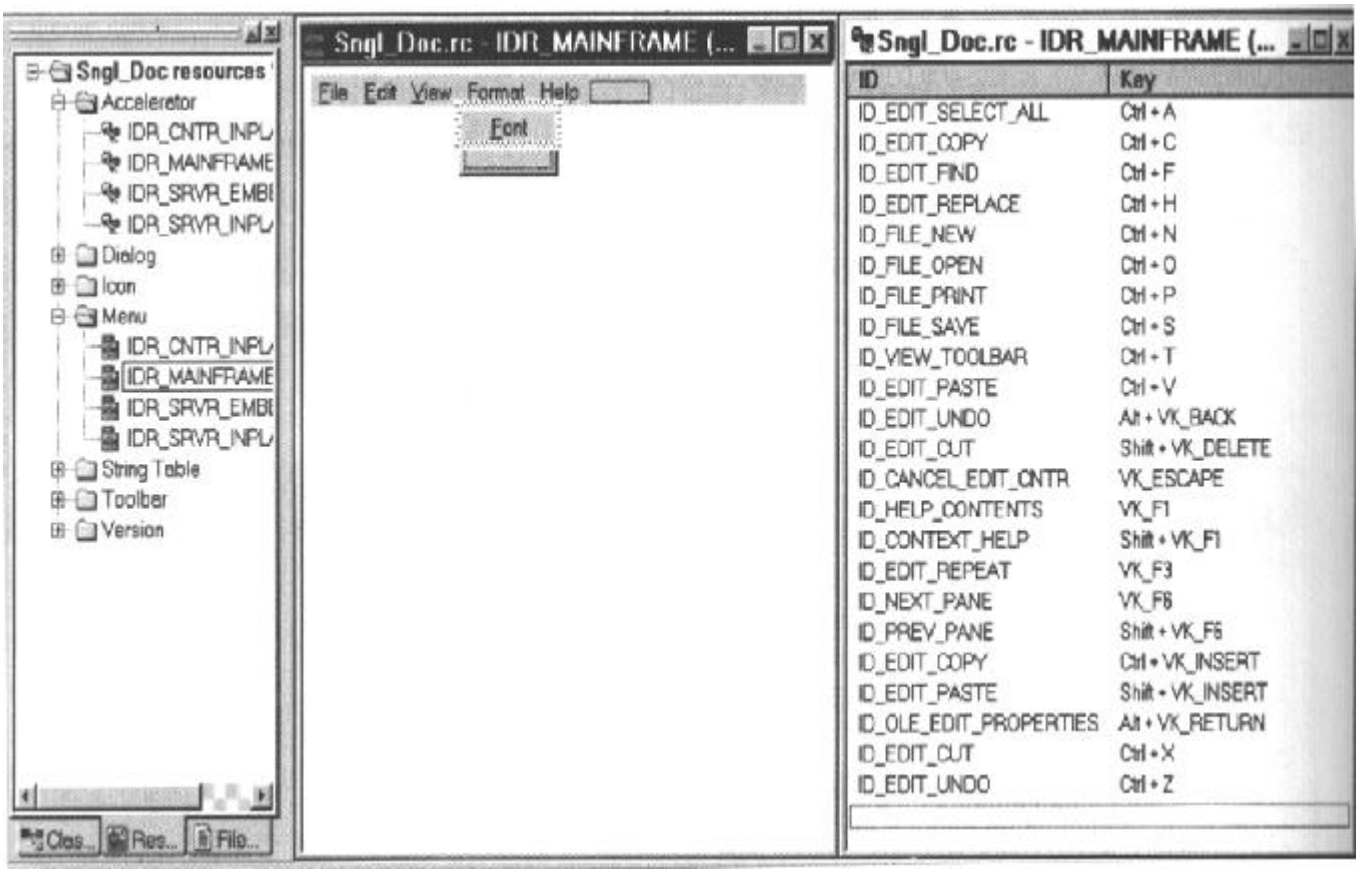


向菜单命令中添加新的文本，这一点很容易。打开 View（视图）菜单，然后右击 Toolbar 条目。从上下文菜单选择 Properties（属性），你会看到如下所示的 Menu Item Properties（菜单项属性）对话框。

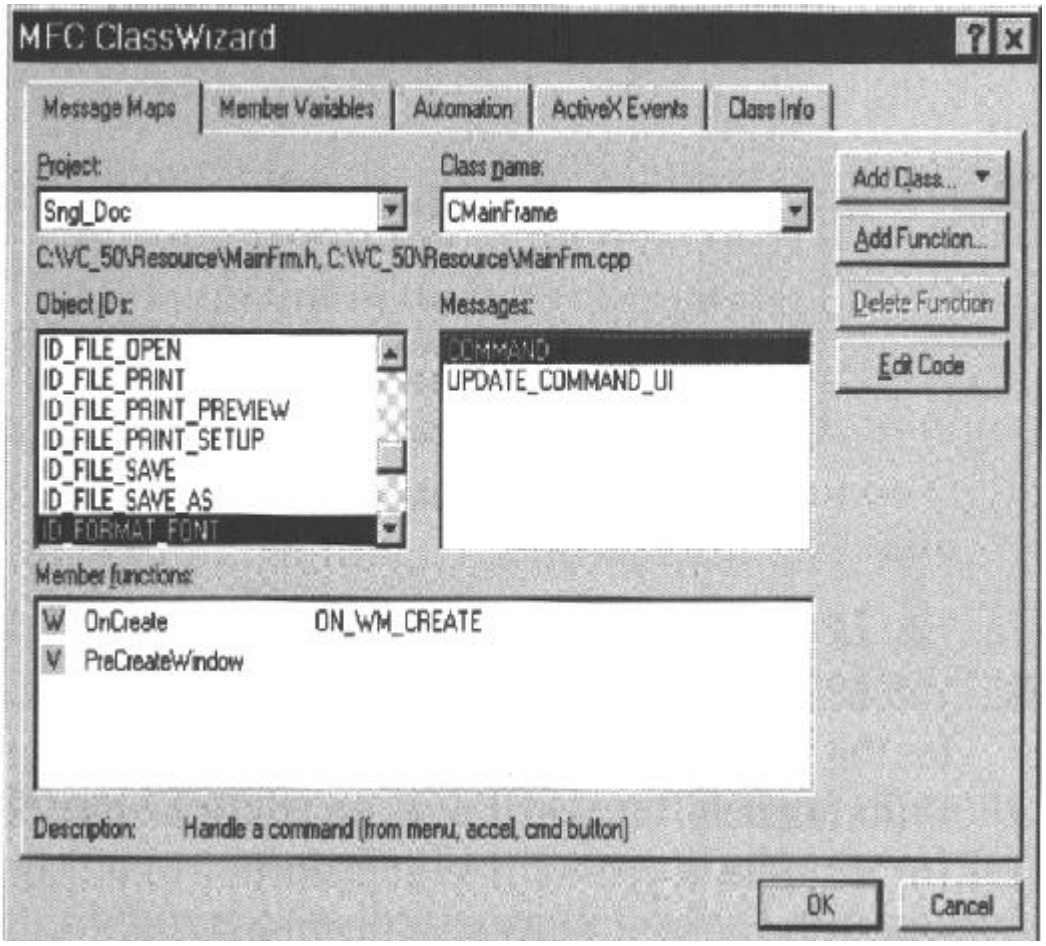
本例中，我们要做的是更改 Caption（标题）域以指明新的加速键。你可以使用通常用于格式化文本的 C 和 Windows 格式化字符。将 Caption 域修改成“&Toolbar\tCtrl-T”，告诉 Windows 你想看到单词“Toolbar”，字母"T"加上下划线，然后是一个空格，最后 CTRL-T 告诉用户该菜单命令的加速键是什么。

那么，如果想添加新的菜单该做些什么呢？只要选择菜单栏或现有菜单上的空白点，开始输入。系统会自动显示一个 Menu Item Properties（菜单项属性）对话框。在本实例中，我们将在工具条中添加一个 Format 菜单，该菜单只有一个菜单项 Font（请记住输入&Format 和&Font，这样每个菜单项的第一个字母才

能加上下划线）。添加了新的菜单项后，将 Format 移到 Help 菜单左侧。此时的菜单如下图所示。



现在我们要添加一些使这个新菜单项发挥作用的代码。右击 Font 菜单项，然后从上下文菜单中选择 ClassWizard，再选择 Message Maps 选项卡。你会看到 MFC ClassWizard 对话框，如下图所示。



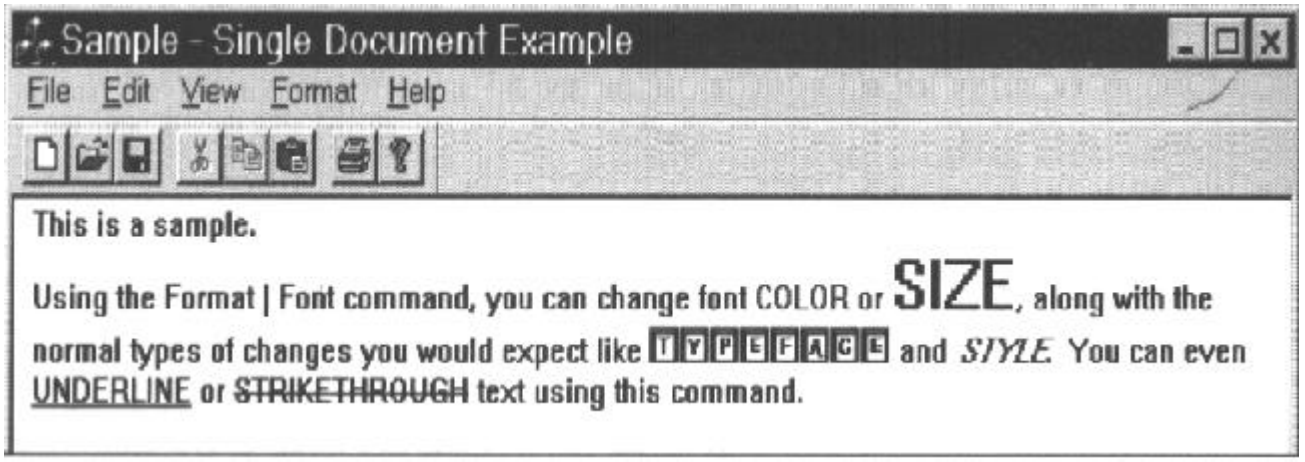
单击 Messages **列表中的** COMMAND**。单击** Add Function**（添加函数）按钮创建这个函数。你会看到** Add Member Function**（添加成员函数）对话框，单击** OK **接受缺省的函数名。单击** Edit Code**（编辑代码）按钮编辑这个新函数。程序列表** 3.3 **显示了向我们的程序中添加字体特性时所需的代码。虽然也可以将其添加到** CEdit **控件中，但只有富文本编辑屏幕（**CRichEdit **控件）才原本就具有这种能力。**

程序列表 3.3

```
void  CMainFrame::OnFormatFont()
 {
      CFontDialog       oDialog;       //Create a font dialog.

      //Display the Font common dialog box.
      oDialog.DoModal();
 }
```

如你所见，在我们的示例中添加处理字体的能力简直太容易了。如果现在编译并运行这个程序，用户就能够改变缺省字体，或选择文本并改变所选文本的字体。下面是利用已经具备改变字体能力的这个示例程序修改字体方式的示例。

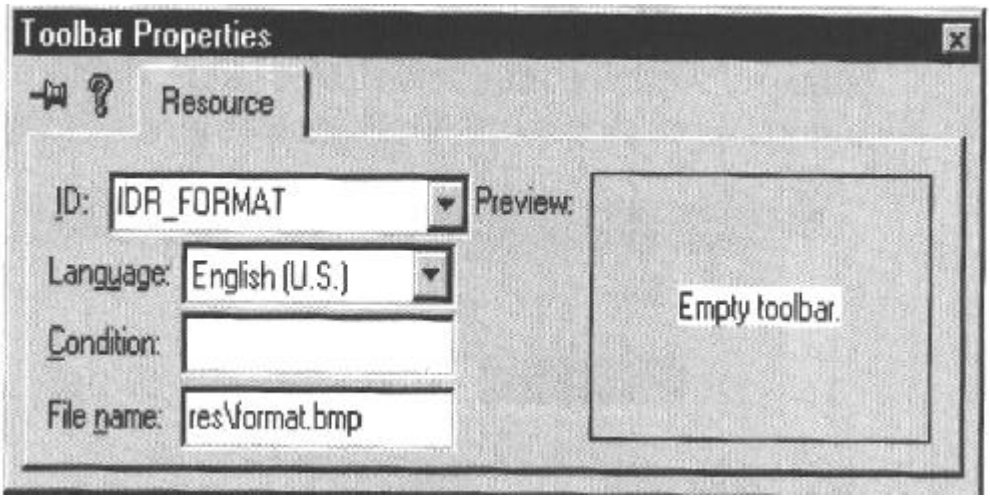


技巧 显示应用程序所需的大部分菜单并不需要做什么艰苦的努力。你真正需要完成的工作，只是运用各种菜单项的准确 ID 和工具条上的相关按钮。可惜的是，许多特殊的 ID 目前尚未列入文档。例如，如果想执行字体对话框而不做任何程序设计，要确保菜单项的 ID 是：ID_FORMAT_FONT。同样地，如果向应用程序中添加工具条按钮，工具条按钮应该与菜单项使用相同的 ID。可以在 MFC\INCLUDE 文件夹的 AFXRES.H 文件中找到所有特殊的 ID，无论它们是否已经列入文档。

3.3 使用工具条

如果说加速键是提高程序访问速度的键盘方法，那么工具条就和鼠标操作对应起来。你会看到，工具条不再仅仅是附属品了，它已经成为用户界面必不可少的一部分。但是，如果使用很多能够满足或者不能满足用户需要的按钮填满整个工具条的话，它很快就会变得过于臃肿。这个问题的一个解决办法是，创建多个工具条，允许用户自己决定当前需要哪些工具条。

使用工具条和使用菜单与加速键一样容易。不过在这里，必须创建工具条及其相关菜单命令之间的链接。缺省工具条 IDR_MAINFRAME 包括一些更为常见的按钮，比如打开文件或创建新文件时用到的按钮。



让我们以创建新的工具条来开始这个示例吧——这个工具条允许用户格式化文本。在 ResourceView 中右击 Toolbar 文件夹，然后从上下文相关菜单中选择 Insert Toolbar（插入工具条）。Visual C++会自动创建一个新的工具条。但是，它给出的名字(IDR_TOOLBAR1)并不具有清晰的意义。右击 IDR_TOOLBAR1 条目，然后从上下文菜单中选择 Properties（属性）。你会看到如下图所示的 Toolbar Properties（工具条属性）对话框。

在 ID 域中输入 IDR_FORMAT（不必担心要修改 File Name（文件名）域；修改 ID 域时它会自动修改）。完成修改后，单击 Close（关闭）框关闭 Toolbar Properties（工具条属性）对话框。

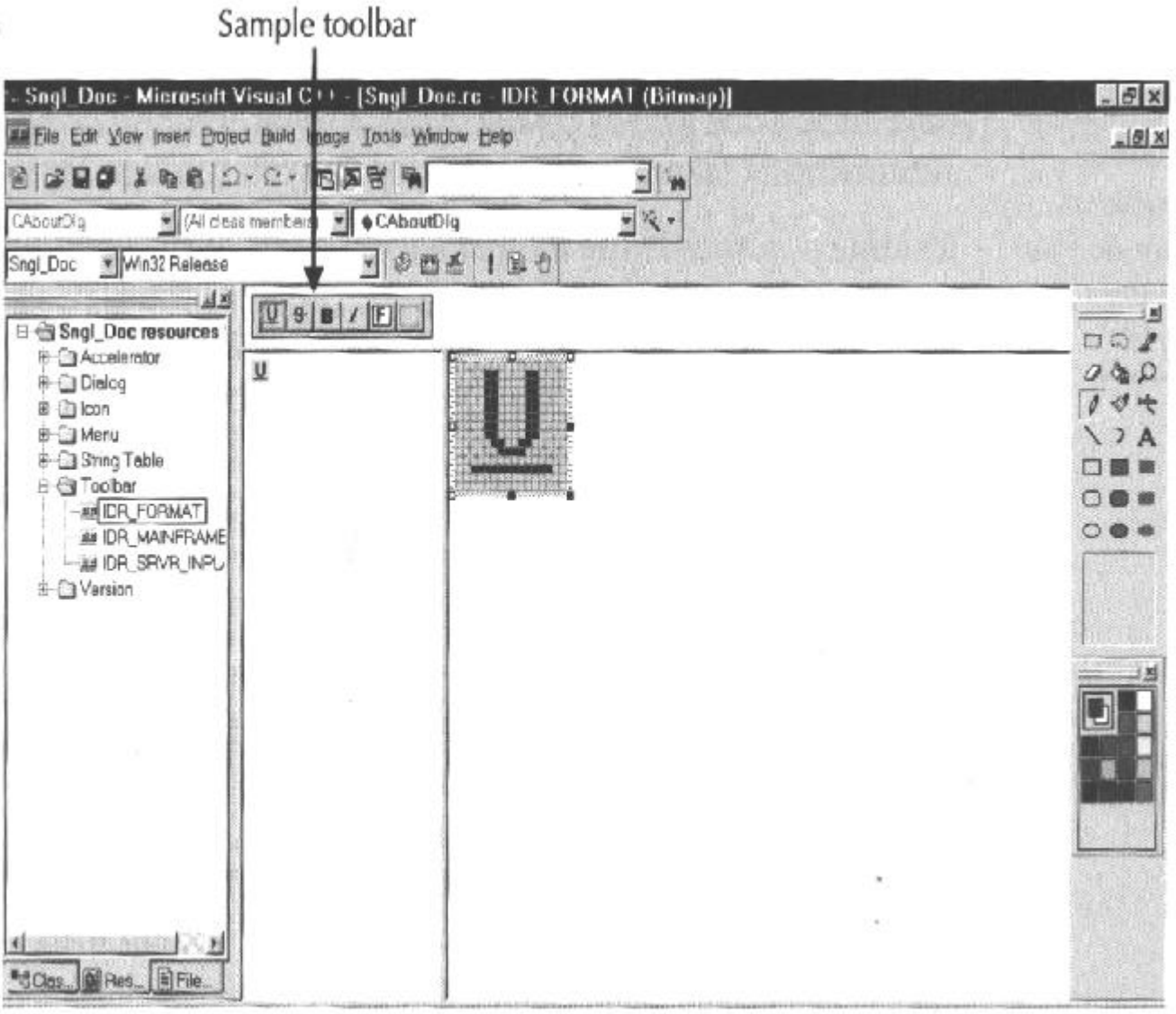
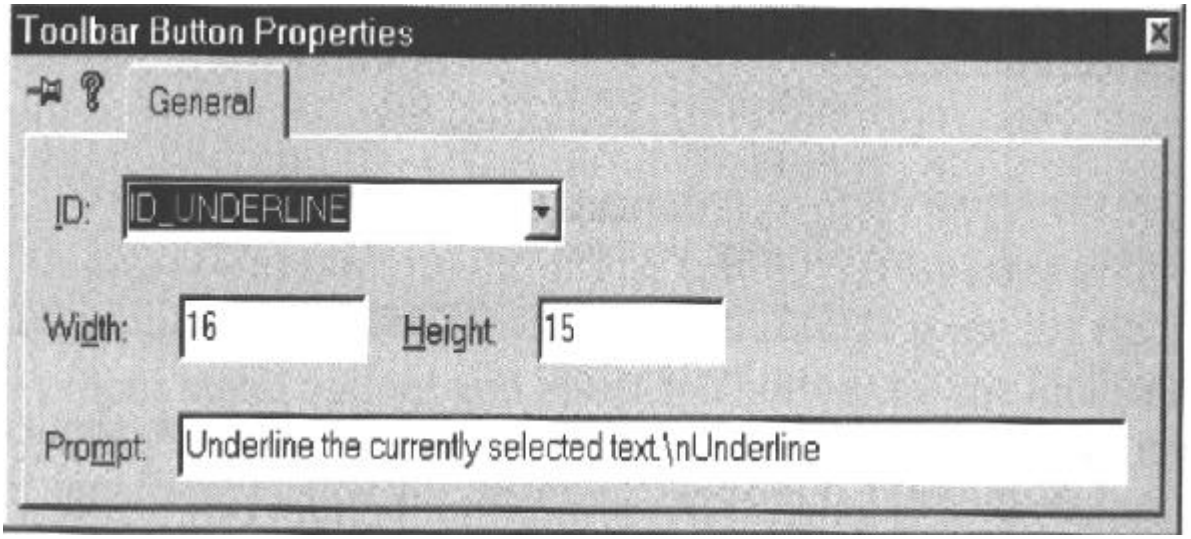


图 3.6　示例工具条上包括了五个按钮

现在我们要向这个工具条中添加一些按钮。这些按钮将允许用户执行各种各样的任务，而不必借助于使用键盘或在菜单系统间移动。图 3.6 显示了我们

将为本示例使用的样本工具条（这些按钮分别代表了加下划线、删除线、粗体、斜体和打开字体对话框）。

向工具条上添加按钮并不需要做太多的工作。你真正需要具备的东西是将来要进行操作的位图。双击下划线按钮，会看到 ToolBar Properties（工具条属性）对话框，如下所示。



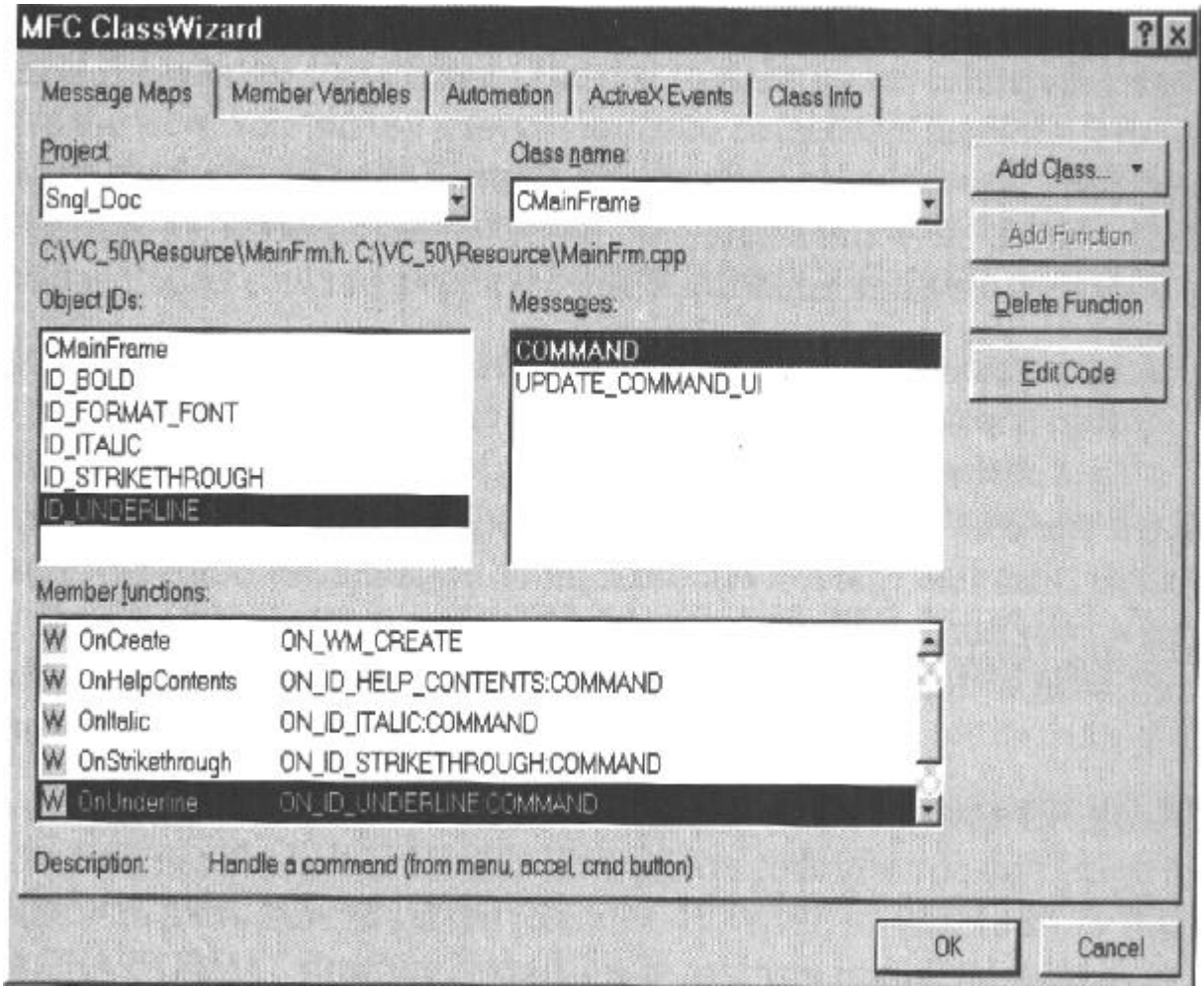
如你所见，我已经把 ID 域修改为 ID_UNDERLINE 了，将 Prompt（提示）域修改为 Underline（下划线）了。用相似的方法修改其它所有的按钮——ID_STRIKETHROUGH（删除线），ID_BOLD（粗体），ID_ITALIC（斜体）和 ID_FORMAT_FONT（字体对话框）。一定要仔细地输入所有的 ID，否则，在今后使用按钮时会出问题。我为最后一个按钮使用 ID_FORMAT_FONT 的原因是，减少所需的代码编写数量。使用这个 ID 意味着，无须添加一行代码就可以让这个按钮发挥作用，Visual C++会通过 MFC 自动处理这个按钮。

注　要了解 MFC 提供的其它标准 ID，请不要忘了浏览 MFC\INCLUDE 文件夹

中的 AFXRES.H 文件。

现在让我们把工具条与应用程序的其它部分联系起来。按住 CTRL 键并双击下划线按钮，你会看到 MFC ClassWizard（类向导），然后系统显示 Adding a Class（添加类）对话框。IDR_FORMAT 是一种新的资源，Visual C++不能确定你想用它做什么。你要把它和现有类联系起来，现在只要单击 OK，接受缺省设置。Visual C++会显示 Select Class（选择类）对话框。选择 CMainFrame 类，然后单击 Select（选择）。你现在已经把工具条和应用程序联系起来了。

创建和按钮相关的函数是很容易的。单击 MFC ClassWizard 的 Object IDS 列表中的按钮，比如 ID_UNDERLINE，然后单击 Messages（消息）列表中的 COMMAND。单击 Add Function（添加函数）按钮创建函数。Visual C++显示 Add Member Function（添加成员函数）对话框。只需单击 OK 接受缺省函数名。最后，你会看到一个成员函数列表，如下图所示。



现在我们可以向程序中添加使按钮发挥作用的代码了。单击任一成员函数名（比如 OnUnderline**），然后单击** Edit Code **按钮。**Visual C++**会显示代码编辑区。你会看到我们刚刚创建的函数的框架。程序列表** 3.4 **显示了需要添加的代码。**

程序列表 3.4

```
void CMainFrame::OnBold()
```

```
{
        CRichEditView*          poView;         // Create a variable to hold our view.

        // Get the active view from the current window.Typecast it as a
        // CRichEditView rather than a CView, the standard return value.
        poView = (CRichEditView*) GetActiveView();

      // Change the font settings as needed.
      poView->OnCharEffect(CFM_BOLD, CFE_BOLD);
}
void CMainFrame::OnItalic()
{
      CRichEditView*          poView;       // Create a variable to hold our view.
      CHARFORMAT              cfFont;       // Create a structure for the font data.

      // Get the active view from the current window. Typecast it as a
      // CRichEditView rather than a CView, the standard return value.
      poView = (CRichEditView*) GetActiveView();

      // Get the current font settings, then change them to Italic.
      cfFont = poView->GetCharFormatSelection();
```

```
        cfFont.dwMask |= CFM_ITALIC;
        cfFont.dwEffects |= CFE_ITALIC;

        // Change the font settings as needed.
        poView->SetCharFormat(cfFont);
}

void CMainFrame::OnStrikethrough()
{
        CRichEditView*          poView;         // Create a variable to hold our view.
        // Get the active view from the current window. Typecast it as a
        // CRichEditView rather than a CView, the standard return value.
        poView = (CRichEditView*) GetActiveView();

        // Change the font settings as needed.
        poView->OnCharEffect(CFM_STRIKEOUT, CFE_STRIKEOUT);
}
void CMainFrame::OnUnderline()
{
        CRichEditView*        poView;         // Create a variable to hold our view.
```

```
        // Get the active view from the current    window. Typecast it as a
        // CRichEditView rather than a CView, the standard return value.
        poView = (CRichEditView*) GetActiveView();

        // Change the font settings as needed.
        poView->OnCharEffect(CFM_UNDERLINE, CFE_UNDERLINE);
    }
```

正如从源代码中所看到的，对选定的一组字符，有两种不同的方法更改字符的字体属性。第一种方法要容易一些，只需得到活动视图——即包含用户正编辑文本的窗口部分。一旦拥有了这个视图，就可以用名为 OnCharEffect()**的特殊函数更改字体属性。要使这个函数真正发挥功能，还要为两个参数提供相同的字体属性（**CHARFORMAT **文档包含了一系列完整的属性及相关定义）。**

第二种方法要求再做一些工作，但同时它也更灵活。这时，必须获得活动视图的一个拷贝。但是这一次，用它把当前字体特性填入 CHARFORMAT**（特性格式）结构。该结构包括你需要知道的一切，比如字体名和颜色，以及字体属性，比如粗体和斜体。一旦获得已填好的** CHARFORMAT**（特性格式）结构，就在屏幕上更改想修改的成员，然后用** SetCharFormat()**函数做实际的更改。**

在很多情况下，你会想用我演示的第一种方法改变字体属性，如粗体和斜体。这样做的话，编码要少得多，并且实现起来也不必非得与结构打交道。但是，在进行更详细的屏幕更改时可以使用 CHARFORMAT**（字符格式）结构，了解这一点对编程工作将会起到重要作用。**

我们已经有了工具条和一些使其发挥作用的代码。我们的示例程序仍然缺少一个重要的特征。如果现在就运行它，会看不到工具条。最后一步是添加菜单项和一些让工具条用起来更方便的代码。我们从菜单项开始。我要做的，只是用和讨论 Format（格式）菜单时一样的过程，向 View（视图）菜单中添加一个新的菜单项。在 Menu Item Properties（菜单项属性）对话框中，我把 ID_VIEW_FORMATTOOLBAR 用作 ID，把&Format Toolbar 用作标题，把显示或隐藏格式工具条的\nToggle Format ToolBar 用作提示。由于我们缺省显示工具条，所以你还要选中 Checked 复选框。

创建程序框架相当容易。按下 CTRL 键并双击新的 Format Toolbar（格式工具条）菜单项。你会看到 MFC ClassWizard（MFC 类向导）对话框。Visual C++ 应该自动加亮 Object Ids（对象 ID）列表中的 ID_VIEW_FORMATTOOLBAR 项。加亮 Messages（消息）列表中的 COMMAND 选项，然后单击 Add Function（添加函数）。最后，单击 Edit Code（编辑代码）显示代码窗口。

有三个地方需要添加工具条代码。第一块代码出现在 MAINFRAME.H 文件中。要在初始工具条变量右下方的 Protected（受保护）节中添加新的变量。新的声明变量的代码如下所示：

```
CToolBar    m_wndToolBar2;
```

下一块代码出现在 MAINFRAME.CPP 文件中（见程序列表 3.5）。这块代码设置工具条的特性，使你在启动程序时能看见它。请注意，为了使工具条成为可“浮动”工具条（这样可以在应用程序中把它从一个地方移到另一个地方），要考虑进行一些特殊的编码。

程序列表 3.5

```
int CMainFrame::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
    if (CFrameWnd::OnCreate(lpCreateStruct) == -1)
        return -1;

    if (!m_wndToolBar.Create(this) ||
        !m_wndToolBar.LoadToolBar(IDR-MAINFRAME))
    {
        TRACE0("Failed to create toolbar\n");
        return -1;                      // fail to create
    }
    if (!m_wndToolBar2.create(this) ||
        !m_wndToolBar2.LoadToolBar(IDR_FORMAT))
    {
        TRACE0(Failed to create toolbar\n");
        return -1;                      // fail to create
    }

    if (!m_wndStatusBar.Create(this) ||
        !m_wndStatusBar.SetIndicators(indicators,
```

```
        sizeof(indicators)/sizeof(UINT)))
{
    TRACE0("Failed to create status bar\n");
    return    -1;          // fail to create
}

// TODO:Remove this if you don't want tool tips or a resizeable toolbar
m_wndToolBar.SetBarStyle(m_wndToolBar.GetBarStyle() |
    CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);
m_wndToolBar2.SetBarStyle(m_wndToolBar2.GetBarStyle() |
    CBRS_TOOLTIPS | CBRS_FLYBY | CBRS_SIZE_DYNAMIC);

// TODO: Delete these three lines if you don't want the toolbar to
// be dockable
m_wndToolBar.EnableDocking(CBRS_ALIGN_ANY);
m_wndToolBar2.EnableDocking(CBRS_ALIGN_ANY);
EnableDocking(CBRS_ALIGN_ANY);
DockControlBar(&m_wndToolBar);
DockControlBar(&m_wndToolBar2);

return 0;
```

```
}
```

有三个需要使用工具条代码的主要区域。第一个主区创建工具条，然后将 IDR_FORMAT 工具条加载进去。如果这一过程失败，那么在应用程序启动前，会得到“创建失败”的消息。第二个主区定义工具条风格。我使用缺省设置，允许用户调整工具条的大小并将其四处移动。鼠标停留在按钮上时，还会显示工具提示。代码的最后一节使工具条拼接化（Docking），定义用户可以在哪里停放工具条，并实际停放我们已创建的工具条。此时，工具条是可见的；用户可以将其四处移动，并用 Close（关闭）框将其从视线清除。

使菜单命令 View | Format Toolbar（视图|格式工具条）发挥作用相当容易。程序列表 3.6 显示了使这一部分程序发挥作用所需的代码。

程序列表 3.6

```
void CMainFrame::OnViewFormattoolbar()
{
        CMenu*      poMenu;             // Create a pointer to the current menu.

      poMenu = GetMenu();           // Get the menu.

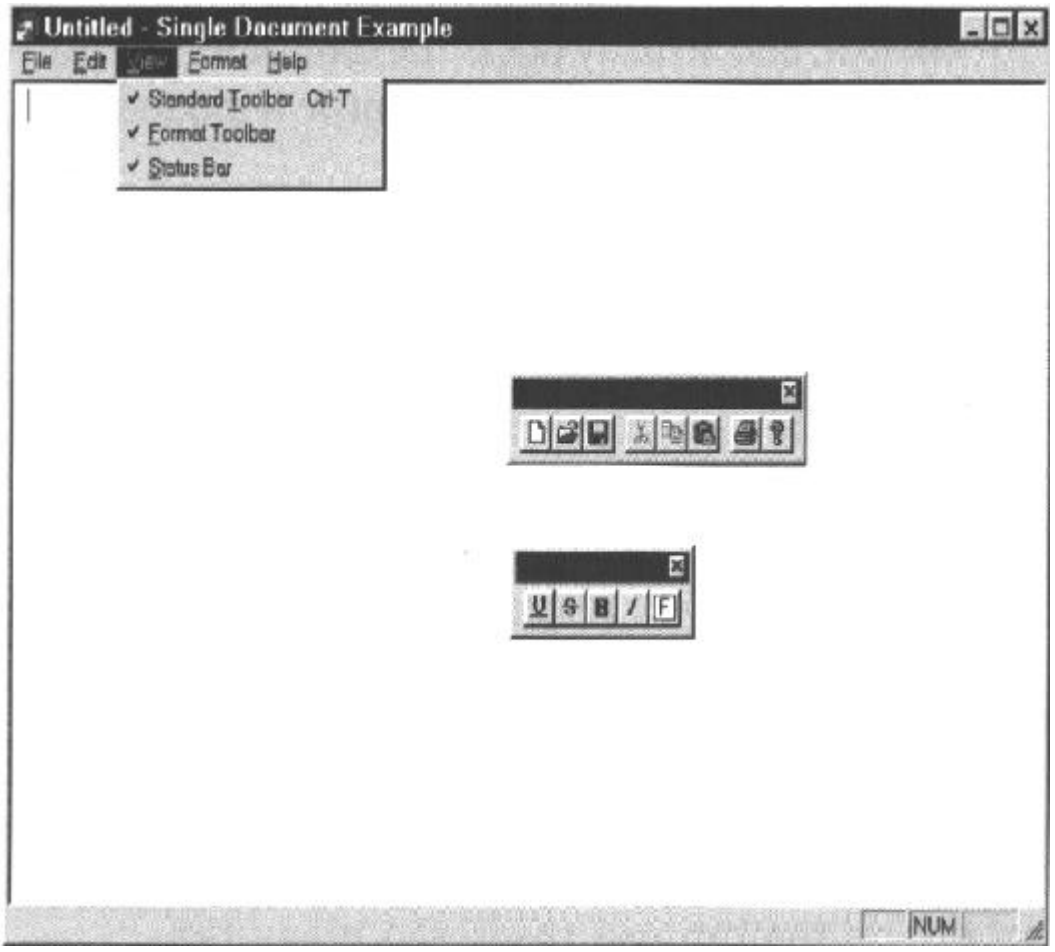
    // Determine if the View | Format Toolbar option is checked. If it
    // is, then hide the format toolbar and uncheck the option. Otherwise,
    // display the toolbar and check the menu item.
```

```
            if (poMenu->GetMenuState(ID_VIEW_FORMATTOOLBAR, MF_CHECKED))
            {
                showControlBar(&m_wndToolBar2, FALSE, FALSE);
                poMenu->CheckMenuItem(ID_VIEW_FORMATTOOLBAR,
MF_UNCHECKED);
            }
            else
            {
                ShowControlBar(&m_wndToolBar2, TRUE, FALSE);
                poMenu->CheckMenuItem(ID_VIEW_FORMATTOOLBAR,
MF_CHECKED);
            }
        }
```

如你所见，我们是从获得 CWnd **类菜单对象的拷贝开始的。当我们得到了这个菜单对象后，要判断当前用户是否选择了** Format Toolbar **（格式工具条）选项就变得很容易了。如果选择了这个选项，那么工具条是可见的。把** ShowControlBar()**函数的第二个和第三个参数设成** false**，可以使工具条不可见。** CheckMenuItem()**允许你从** View | Format Toolbar **（视图|格式工具条）菜单选项去掉复选标记。反过来，用相反的过程使工具条可见并再次选择菜单选项。**

继续下去，再编译一次应用程序，这样你才能熟悉我们刚刚添加的各种特

征。一定要试验所有的格式化选项和连接及隐藏工具条的功能。很明显，这个应用程序并不像你现在看到的一些程序那样复杂，但它确实可以充分地使用资源。下面是我的程序版本的外观，其中工具条尚未连接在一起。



第二部分

Visual C++与数据库管理



返回总目录

目　　录

第 4 章　ADO 与 ODBC 4

4.1　什么是 ODBC 8

4.2　ADO 和 OLE-DB——同一问题不同层次的解决方案 24

4.3　什么时候应该使用 OLE-DB，ADO 或 ODBC 29

4.4　Visual C++ ODBC，OLE-DB 及 ADO 类概述 32

第 5 章　数据库创建概述 35

5.1　构造块概述 37

5.2　创建数据库 55

5.3　添　加　表 59

5.4　使用查询排序数据 74

5.5　创建测试窗体 78

第 6 章　使用 ODBC 建立数据库应用程序 82

6.1　创建简单的窗体视图应用程序 85

6.2　操纵数据库的内容 112

第 7 章　使用 OLE-DB 或 ADO 建立数据库应用程序 146

7.1　**创建简单的网格视图应用程序** 149

7.2　**向应用程序添加报告** 178

第4章　ADO与ODBC

有很多种使用数据库的方法，对大多数数据库来说，选择 C++这种产品也许并不适宜。我们知道，像 dBASE IV，FoxPro，Oracle 和 Access 这样的产品是完全以数据库管理为中心的。事实上，这些产品非常善于创建数据库管理器，以至于它们确实并不善于做太多其它的工作。即使要用更通用化而非更专用化的数据库产品来执行一些类型的工作，在使程序设计更容易这一方面，像 Visual Basic 和 Delphi 这样的 RAD 环境也要比 Visual C++强很多。

你是不是对我的说法感到很奇怪？下面我就要谈一谈，在谈到使用数据库管理系统(DBMS)这个话题时，用 Visual C++实际上可以做些什么。虽然上述其它语言使得编写成熟的包括用户界面和高速搜索能力的 DBMS 就像孩子做游戏一样容易，但是，它们缺少 Visual C++可以提供的某些重要东西。你不能为使用 Access 的数据库轻松地编写出实用程序。正像实用程序的定义所说的，实用程序应该很小并且具备可移植性——Access 应用程序却不是这样。即使用 Access 这样的产品创建的程序可以很小并且可以移植，你仍有其它方面的需求：底层的功能。

注：编写数据库实用程序及驱动程序时，可以选择 Visual C++语言。

想像一下，使用像 Visual Basic 这样的语言来与实时数据采集设备打交道的情况。在进行底层访问时，RAD 的保护环境常常使程序员不能进行有效的处理。当然，数据采集设备几乎不依赖于简明的连接。你打算如何把 Visual Basic 和外

部的数据源连接起来呢？数据源甚至可能不了解 Windows，DOS 或类似的成熟的操作系统。

只要使用得当，很容易看到 Visual C++是一种不可或缺的数据库管理工具。针对大规模的应用程序，即使你仍想依赖于 Visual Basic 这样的 RAD 语言，也请考虑一下 Visual C++，它创建的程序规模小、提供底层访问并能提供实时访问。事实上，你可能还没有想到，Visual C++数据库应用程序的市场是很有潜力的。随着人们在旅途中越来越多地使用膝上型和掌上型电脑，这两类电脑上的数据库应用程序也变得越来越普通。你也许能够适应今天的膝上型电脑上的 Access 应用程序，但谈到硬盘大小或内存需求时，公司里较老的膝上型电脑可能就达不到要求。运行 Windows CE 的掌上型电脑在运行这个 Access 应用程序时，肯定会发生故障。在这一数据库市场的新领域，Visual C++提供了无价无限的工具。

Web 链接 谈到使用 Visual C++和数据库，其实你并不孤单。从一开始就有数据库专用新闻组提供有关数据库创建技巧的帮助，比如 microsoft.public.access。不过，这些新闻组提供的是通用信息，对实际编写应用程序并非全都那么有用。专门针对 Visual C++问题的新闻组是 microsoft.public.vc.database 和 microsoft.public.vc.mfcdatabase。如果你决定用 ODBC 访问数据库，可能还要查看一下 microsoft.public.odbc.sdk 新闻组，它讨论的不仅仅是 SDK。对最新技术感兴趣的程序员可以查阅 microsoft.public.ado 新闻组，或者 microsoft.public.oledb（对

象链接和嵌入数据库）新闻组，前者讨论 ADO，后者讨论 ADO 的基础技术。在 microsoft.public.ado.rds 有一个 ADO 子组，它讨论远程数据访问。

既然所有的疑惑都消除了，大多数人的信心也就增强了，下面我们就介绍两种使 C++访问数据库中的数据的主要方法：ODBC（开放数据库互连）和 ADO（ActiveX 数据对象）。在本章中，将介绍这两种类型的访问方法，但我想你会发现，ADO 方法是针对新的程序设计情形而采用的。它克服了早期技术的诸多限制，依赖于 Microsoft 新的底层访问方法 OLE-DB（对象链接和嵌入数据库）。在本书的后面我们会看到，用 ADO 和 Visual C++提供的各种向导来汇集数据库工程，其速度有多快。

注　ODBC 通常用来访问不具备 OLE-DB 特性的非 Microsoft 数据库中的数据；16 位的 ODBC 驱动程序工作起来可能非常缓慢。

ODBC 素以最慢的数据访问方法而著称，但是很可惜，当 ADO 或 DAO 都不支持某个数据库管理器而 ODBC 支持这个数据库管理器时，在这种特定的情形下，你仍然需要使用 ODBC。在大多数情况下，这意味着要从数据库厂商那里获得所需的驱动程序，虽然 Visual C++确实附带了一些产品的驱动程序（如果你正在使用数据库管理器的某些神秘功能，那么就需要建立自己的接口——这并不是一件十分困难的事）。本质上讲，你总是要使用 ODBC 来访问 Microsoft 产品之外的其它 DBMS 产品所创建的数据库，这些数据库并不具备 OLE-DB 功能。ODBC 还要求做一些额外的工作——为 ADO 调整 Visual C++中的大部分向导。

高级技巧

除了使用 ADO 和 ODBC 外，你还可以使用像 DAO（数据访问对象）这样的早期技术，该技术包含在像 Access 这样的 Microsoft 产品中。DAO 依赖于用 Microsoft Access 自动获得的 Microsoft Jet 数据库引擎。DAO 还是较早版的 Visual Basic 所使用的引擎（最新版的 Visual Basic 和 Visual C++依赖于相同的 ADO/OLE-DB 组合），所以如果需要支持较早的 Visual Basic 应用程序，那么 DAO 仍是一个不错的选择。

尽管 Microsoft 文件声明，可以用 DAO 访问非 Microsoft 产品建立的数据库，但你仍会发现，在这种情况下，使用 ADO 和 ODBC 要好得多。这样的话，不但兼容性问题会少一些，速度也将有所提高，因为数据请求经过的接口层减少了。有一条经验要记住，DAO 是设计用来处理 MDB 文件的。

DAO 的一个问题是，它不支持远程通信。这是 Microsoft 提出 RDO（远程数据对象）的原因之一。这种特别技术在 Visual Basic 应用程序中的使用，要比在 Visual C++中的使用多得多，所以我猜想，你们中有很多人都在使用它。但是，记住 RDO 仍是一种生命力很强的技术，这一点很重要。ADO 确实具有替代 RDO 的远程数据服务(RDS)特征。换言之，ADO 在一个软件包中提供了 DAO 和 RDO 两种功能性。

4.1 什么是ODBC

开放数据库互连（ODBC)是 Microsoft 引进的一种早期数据库接口技术。它实际上是我们要在本章后面加以讨论的 ADO 的前身。Microsoft 引进这种技术的一个主要原因是，以非语言专用的方式，提供给程序员一种访问数据库内容的简单方法。换句话说，访问 DBF 文件或 Access Basic 以得到 MDB 文件中的数据时，无需懂得 Xbase 程序设计语言。事实上，Visual C++就是这样一个程序设计平台，即 Microsoft 最初是以 ODBC 为目标的。

你会发现，ODBC 工作起来和 Windows 一样——它用包含在 DLL 内的驱动程序完成任务。其实，ODBC 提供一套两个驱动程序：一个是数据库管理器的语言，另一个为程序设计语言提供公用接口。允许 Visual C++用标准的函数调用经公用接口访问数据库的内容，是这两个驱动程序的汇合点。当然，还有其它和 ODBC 有关的实用程序类型的 DLL。例如，一个这样的 DLL 允许你管理 ODBC 数据源。ODBC 的实际管理接口出现在 SYSTEM 文件夹中的某个 CPL（控制面板）文件中——我们在后面要谈到这方面的问题。

ODBC 的确能履行承诺，提供对数据库内容的访问，并且没有太多的问题。它没有提供数据库管理器和 C 之间尽可能最好的数据转换，这种情况是有的，但它多半能像广告所说的那样去工作。唯一影响 ODBC 前程的是，它的速度极低——至少较早版本的产品是这样。ODBC 最初面世时，一些开发者曾说，因为速度问题，ODBC 永远也不会在数据库领域产生太大的影响。然而，以 Microsoft 的市场影响力，ODBC 毫无疑问是成功了。今天，只要有两种 ODBC 驱动程序

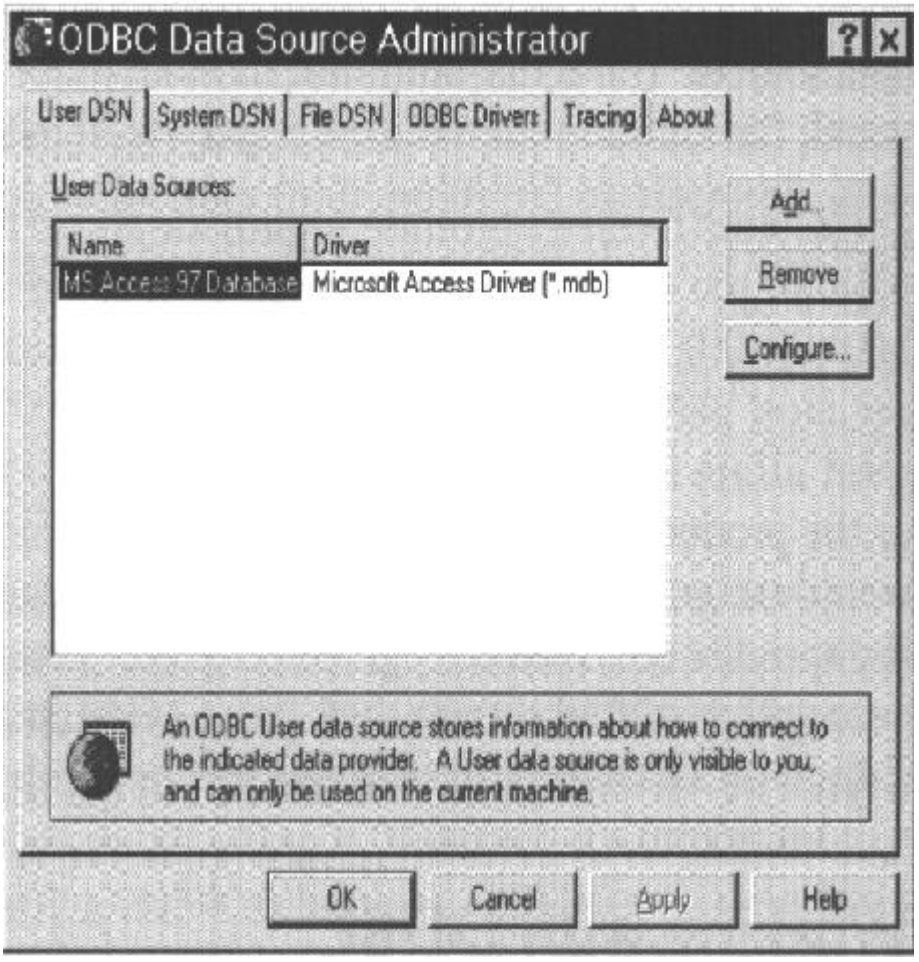
的一种，那么几乎每一个数据库管理器的表现都会很卓越。

使用 ODBC

在可以用 ODBC 做任何事之前，必须有一个数据库——至少在脑子里。在一个像 Access 这样的应用程序中创建数据库框架通常会更容易一些，因为 Access 可以非常轻松地提供完成任务所需的各种功能。但是，如果你至少安装了所需的 ODBC 驱动程序，那就可以在 C++中创建程序框架。一设计了数据库，就要为其创建 ODBC 数据源，我们会在本节中看到这一点。下面的过程并没有严格地说明活动的经过，只是说明了一种配置数据源的技术。

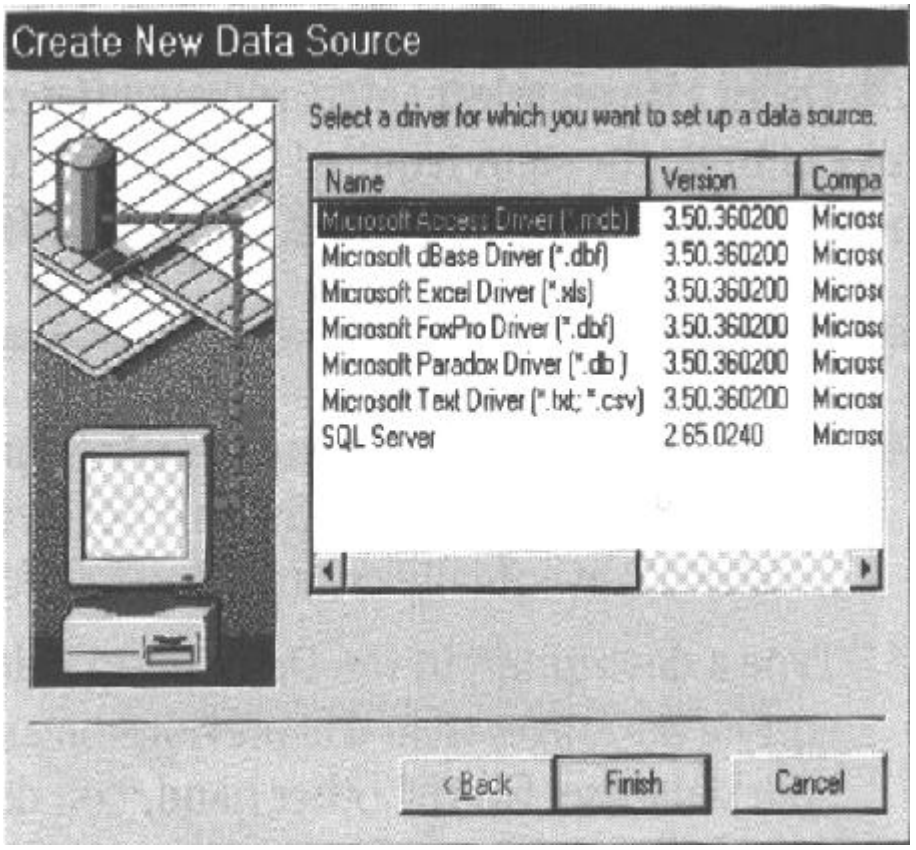
注释 我会在第 5 章中说明如何创建一个 Access 数据库。现在，我们只是看一看，为了访问那个数据库，我们下一步要做些什么。

1. 双击 Control Panel（控制面板）中的 32 位 ODBC 小程序（如果当前系统上未安装 16 位驱动程序，那么一些版本的 Windows 使用简单的 ODBC 小程序）。你会看到 ODBC Data Source Administrator（ODBC 数据源管理员）对话框，如下图所示。请注意，该对话框中当前显示的选项卡是 User DSN（用户数据源名）选项卡。对系统级数据库来说，使用 System DSN（系统数据源名）选项卡；对文件级数据源（从严格意义上说，它不是数据库），使用 File DSN（文件数据源名）选项卡。你还会看到一个数据库的缺省选项。如果想从 Visual C++内部创建数据库，就会用到这个选项。



警告 对本地数据库来说，通常要在 User DSN（用户 DSN）选项卡上创建一个项；对远程数据库，则在 System DSN（系统 DSN）选项卡上创建。任何情况下，都不能在 User DSN（用户 DSN）和 System DSN（系统 DSN）选项卡上创建同名的项。通常会出现的问题是，你试图访问远程数据库，但从 Web 服务器获得了非常奇怪和矛盾的错误消息。事实上，如果在远程数据库访问期间得到了奇怪的错误消息，那么 32 位 ODBC 小程序是首先应该查看的地方之一。

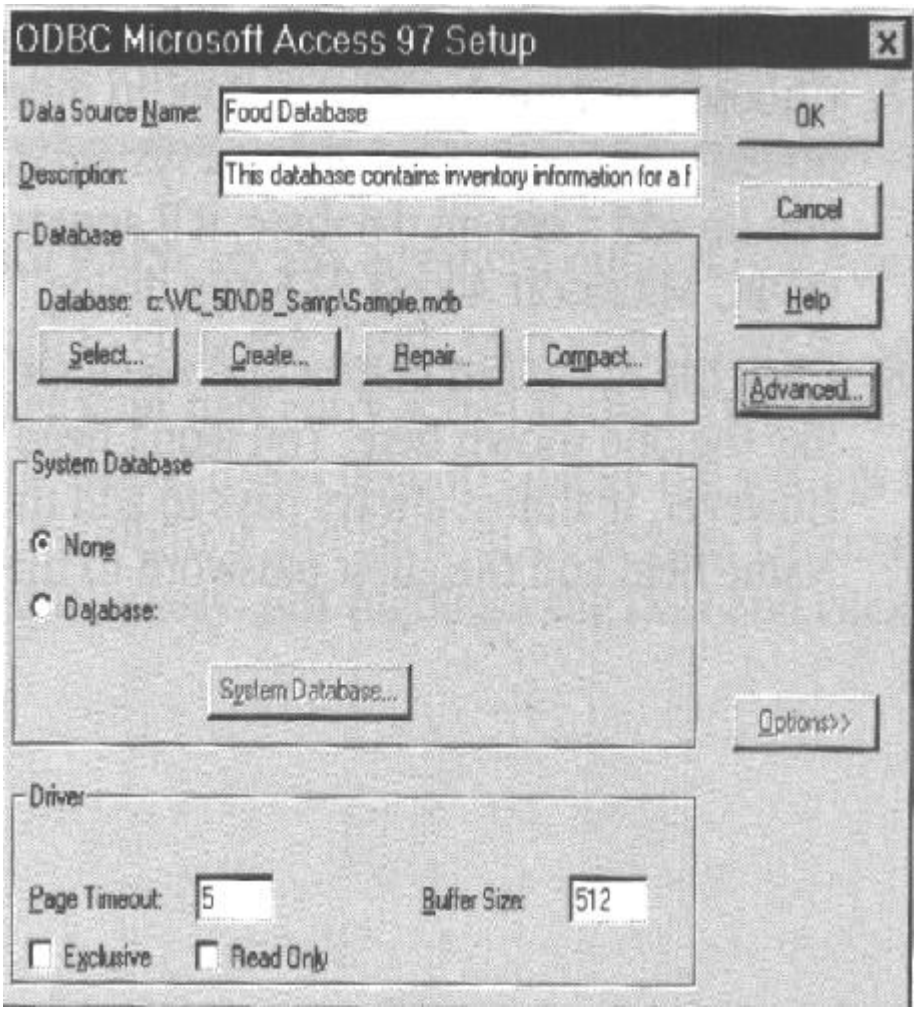
2. **单击** Add **按钮。会看到** Create New Data Source **（创建新的数据源）对话框，如下图所示。**



技巧 检查 ODBC Data Source Administrator（ODBC 数据源管理员）对话框的 About 选项卡，可以确定你正在使用的 ODBC 驱动程序的最新版本。这个选项卡包含了各种 ODBC DLL 的版本号、生产厂商的名称以及出现在 SYSTEM 文件夹中的文件名。大多数情况下，通过查看版本号可以验证 ODBC 驱动程序是否是最新的版本。

3. **选择一个数据源。对本练习来说，我选择了** Access **数据源。单击** Finish **（完**

成），会看到某种类型的配置对话框，如下面的 ODBC Microsoft Access 97 Setup（设置）对话框所示。



注释 如果你选择的数据源和我在本实例中选择的不同，那么所需的配置步骤也和这里说明的不同——每个 ODBC 驱动程序都要求不同类型的配置。

4. 在 Data Source Name（数据源名）域内输入数据源名称。一定要选择意

义明确但又不过于冗长的名称。我选择 Food Database（食品数据库），因为我最终要创建一个与食品库存有关的数据库的链接。

5. 在 Description（描述）域内输入一段说明性文字。可以让这个项比上一个项稍长一些，因为它描述数据库的用途。另一方面，也不要写入像《战争与和平》那样大的小说。对本练习，我输入了：This database contains inventory information for a food store（本数据库包含食品存储的库存信息）。

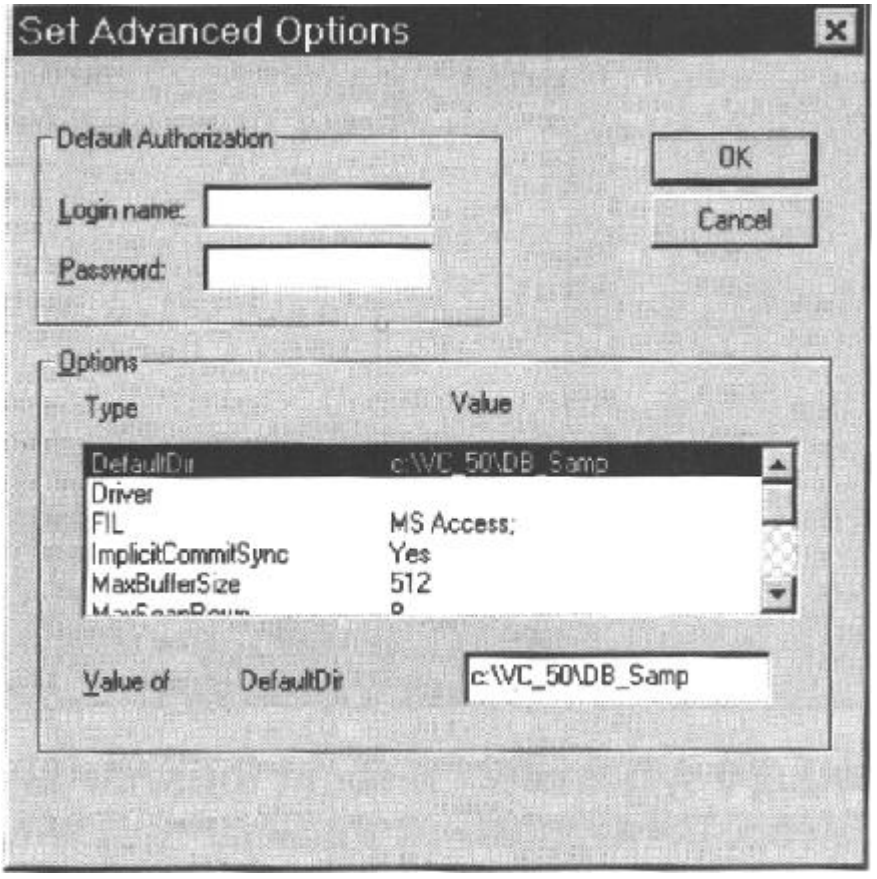
6. 单击 Select（选择）按钮。你会看到一个 File Open-type（文件打开类型）对话框，可以在那里选择一个现有的数据库。ODBC 驱动程序会自动选择正确的文件扩展名。

技巧 并不是一定要提前设计数据库。请注意，Access ODBC 驱动程序还包括一个创建新数据库的按钮。很多 ODBC 驱动程序都提供了这种功能，但并不是全都这样。单击这个按钮会启动数据库管理器应用程序，并允许你设计数据库。注意有一点很有意思，Access ODBC 驱动程序还会允许你使用这个对话框压缩或修补数据库。

7. 选择系统数据库选项。在大多数情况下要选择 None（无），除非你为应用程序特别创建了一个系统数据库。如果确实添加了系统数据库，它会出现在 ODBC Microsoft Access 97 Setup（设置）对话框的 System DSN（系统 DSN）选项卡上。

8. 单击 Advanced（高级）按钮，会看到 Set Advanced Options（设置高级选项）对话框，如下图所示。无需对很多项做修改。但是，要把客户名添加到 Login Name（注册名）域中，把客户口令添加到 Password（口令）域中。这允许客户

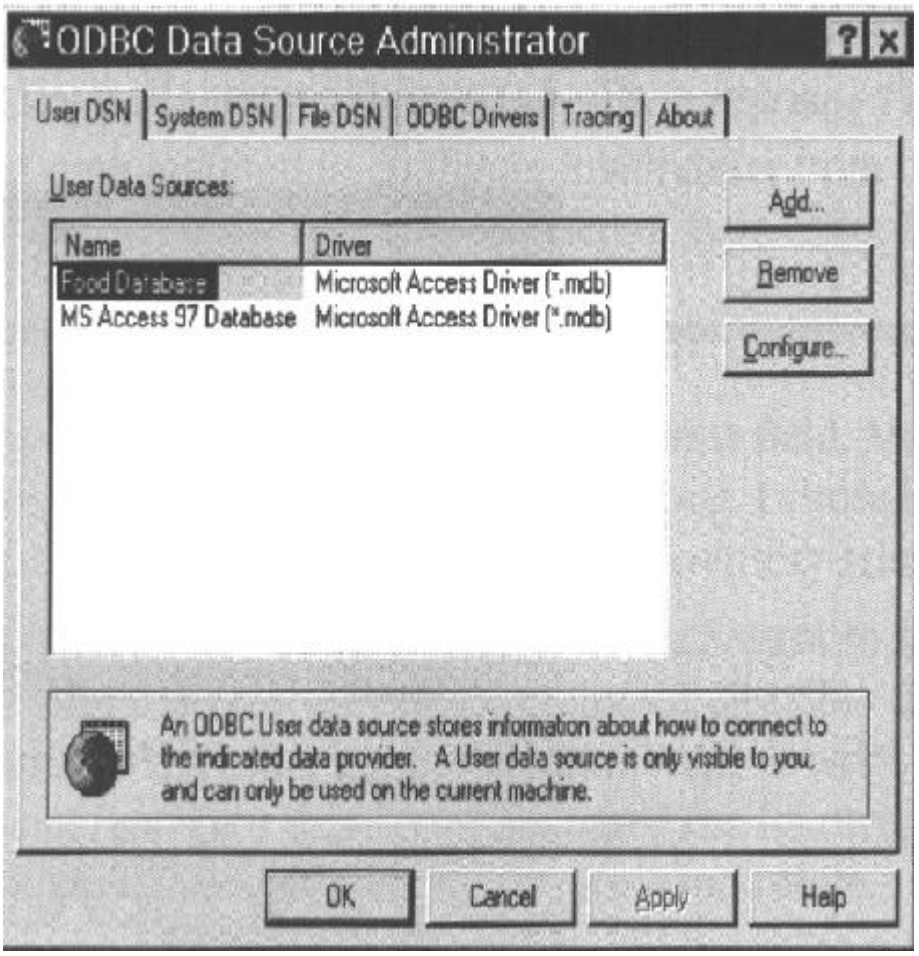
在访问你的数据库时，根本不用了解访问的细节——甚至不用了解被记录的客户名。



技巧 为最优化潜在的区域，可以浏览一下 ODBC 驱动程序提供的一系列高级选项。例如，Access ODBC 允许你更改 DBMS 所用的线程数量。缺省设置 3 通常提供了不错的性能，但是你会发现，复杂程序中线程多一些的话，可以提高前台任务的速度。由于 Windows 使用了一些处理器循环对线程实施管理，所以使用了过多的线程又会降低应用程序的速度。

9. **设置完所需的高级选项后单击** OK。

10. **再次单击** OK **关闭** ODBC Microsoft Access 97 Setup **（设置）对话框。应该看到，新的设置项已经添加到** ODBC Data Source Administrator **（**ODBC **数据源管理员）对话框中。如果今后要为数据库更改这些设置，只要简单地加亮它并单击** Configure **（配置）。删除数据库配置也很容易，只要加亮** DSN **并单击** Remove **（删除）即可。**



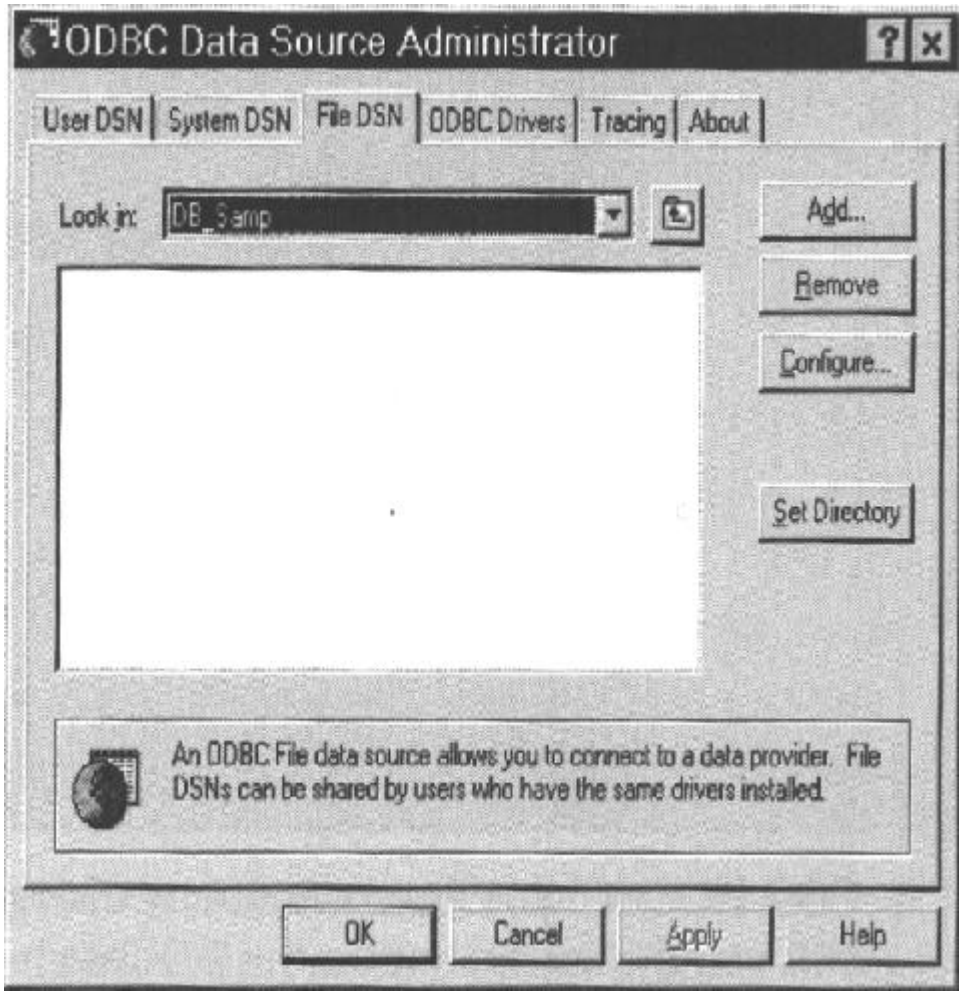
创建系统 DSN **和制作用户** DSN **差不多。两者间一个显著的差别是，使用**

它们的目的不同。系统 DSN 告诉应用程序如何与数据库相连，在一些情况下如何与之交互作用。系统 DSN 不包含数据库所需的任何数据——它包含连接标准，其中可以包括从用户列表到重要文件位置的一切信息。

文件 DSN

你可能已经注意到了，上一节中的实例有问题。如果想单独配置网络上的每一台机器，它会工作得很好，但这可能不是一个好主意。还有一种存储创建数据源所需信息的方式：文件 DSN。这正是我们要在本节中讨论的。下面的过程说明了设置文件 DSN 的一般方法。

1. 双击 Control Panel（控制面板）中的 32 位 ODBC 小程序，会看到 ODBC Data Source Administrator（ODBC 数据源管理员）对话框。选择 File DSN（文件 DSN）选项卡，会看到如下图所示的对话框。要做的第一件事是，选择存储 DSN 信息的地方。

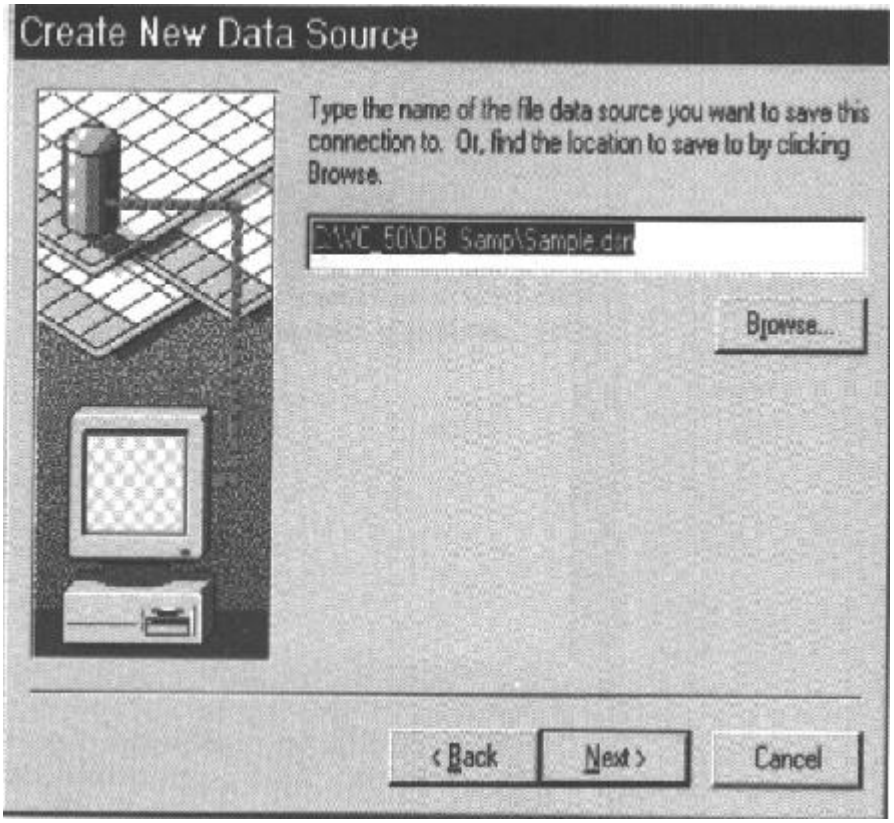


2. **单击** Look In **（观察）下拉列表框，会看到一系列当前机器的目录和驱动器。可以为** DSN **使用任何存储位置。我通常在网络上选择数据库存储目录。使用** UNC**（通用命名标准）意味着，每个人都会用相同的路径来访问** DSN **文件。**

技巧 Up One Level（上一级）按钮（在 Look In 下拉列表框旁边）的工作方式和在 Explorer 中是一样的。可以用该按钮返回到上一级目录。最后，你会在 My Computer（我的电脑）处结束，看到机器上所有驱动器的清单。

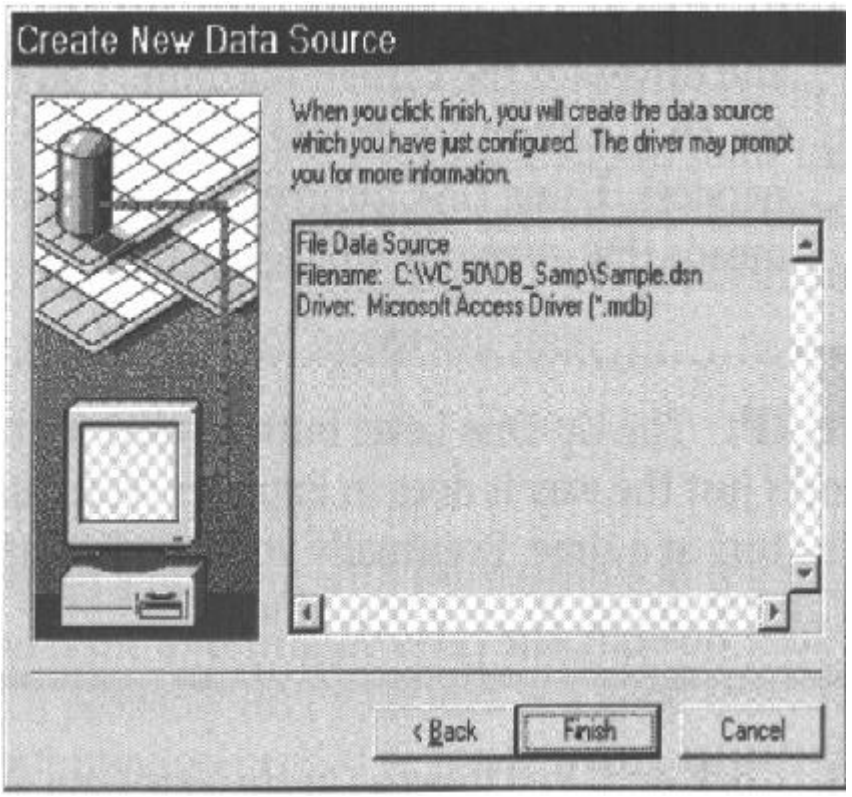
3. **单击** Add（**添加**），**会看到** Create New Data Source（**创建新的数据源**）**对话框。**

4. **在列表中选择一个** ODBC **驱动程序，然后单击** Next（**下一个**）。**对本示例来说，我再次选择** Access。**你会看到** Create New Data Source（**创建新的数据源**）**对话框的下一页显示出来。在这里选择数据源的名称和存储位置。单击** Browse（**浏览**）**会看到** File Open-type（**文件打开类型**）**对话框，在这里选择存储位置。输入一个文件名，**ODBC **向导自动添加** DSN **作为扩展名。在本示例中，我选择** SAMPLE.DSN **作为** DSN **文件的名称。**

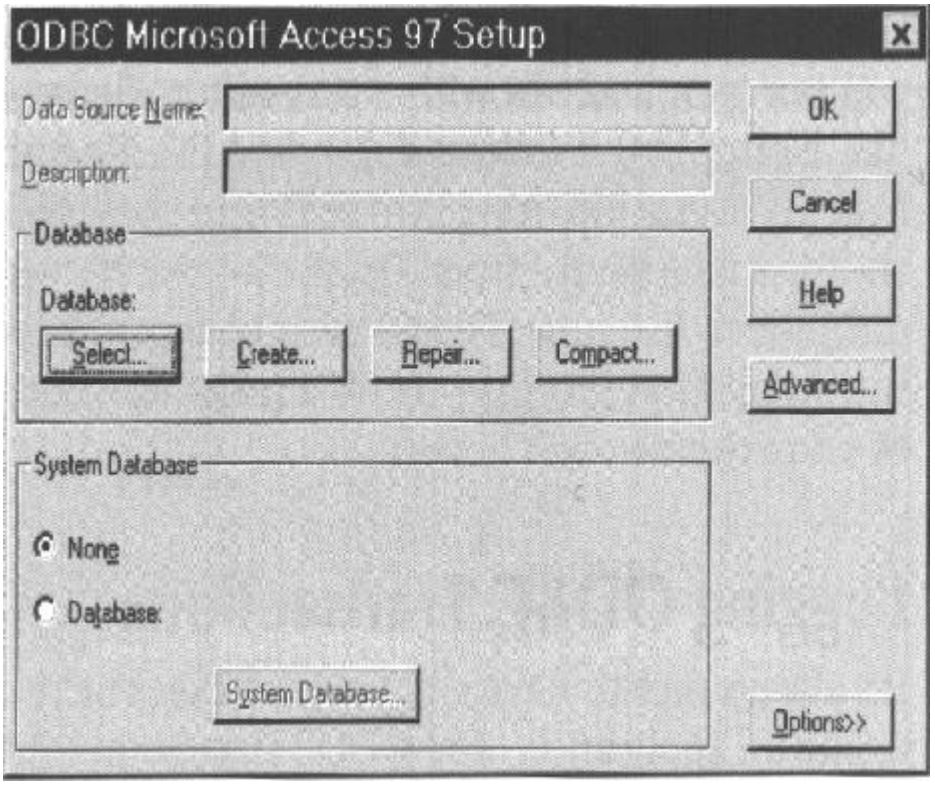


5. **单击** Next（**下一个**）**会看到一个摘要对话框，如下图所示。它说明正准**

备创建的 DSN **的参数。**

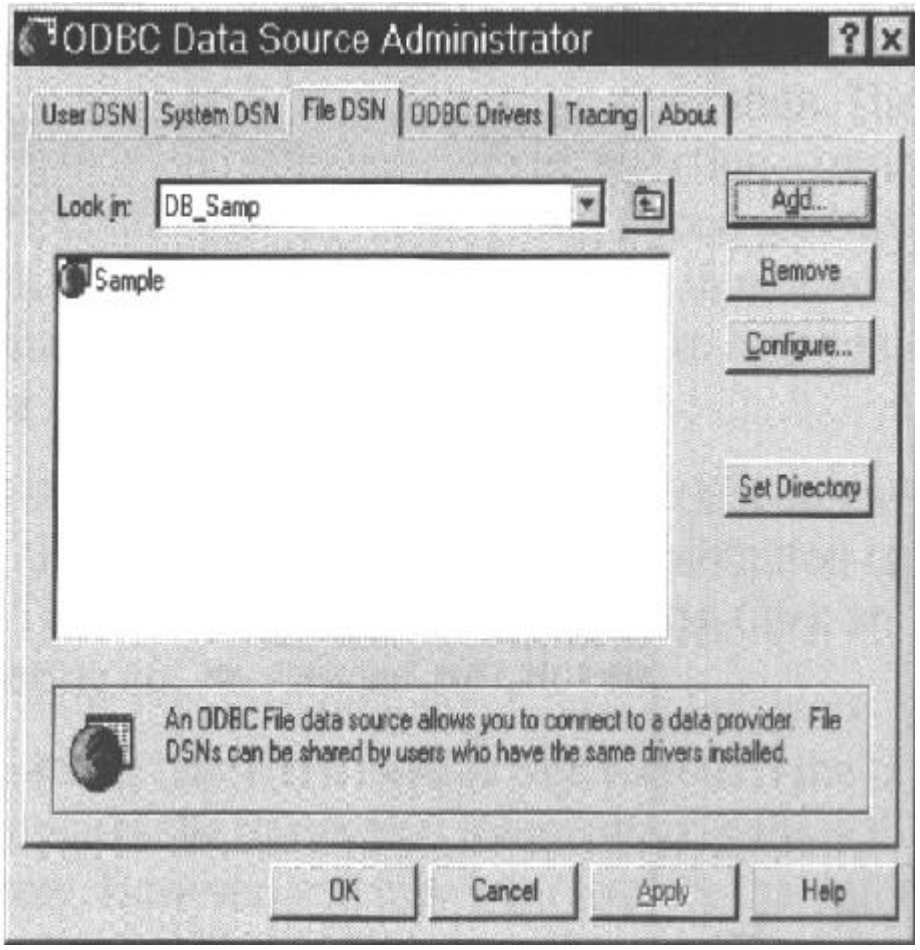


6. **单击** Finish**（完成）。这时，会看到** ODBC Microsoft Access 97 Setup**（设置）对话框的修改版。不能像我们在上一节所做的那样，在** Data Source Name**（数据源名）或** Description**（描述）字段中添加信息了，不过，其它的都和以前一样。**



7. **一定要单击** Select（**选择**）**按钮输入数据库的名称，然后选择想要使用的数据库（如果想创建一个新数据库，也可以单击** Create（**创建**））**。**

8. **完成配置过程时单击** OK，**你会在** ODBC Data Source Administrator（ODBC **数据源管理员）对话框中看到一个新的** DSN **文件项。**



和我们创建的前一个 DSN **不同，这个** DSN **实际上创建可以用文本编辑器来查看和编辑的文件。图** 4.1 **显示了我的文件的外观。请注意，它符合标准的** INI **文件格式。你可以在顶端看到**[ODBC]**标题。下面是我选择的全部设置。这个文件允许我从** Visual C++**选择数据源，而机器间的传输也是非常容易的。我甚至能在安装过程中按要求更改位置——在你不知道用户会有什么样的设置时，这一点确实很不错。**

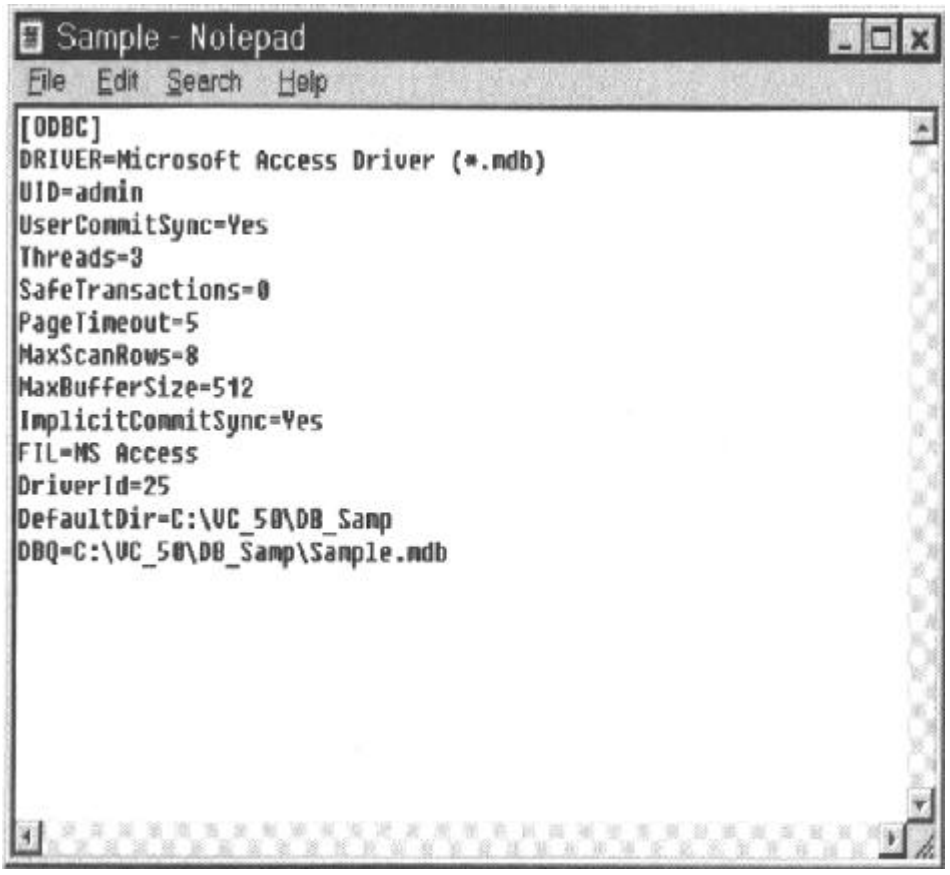


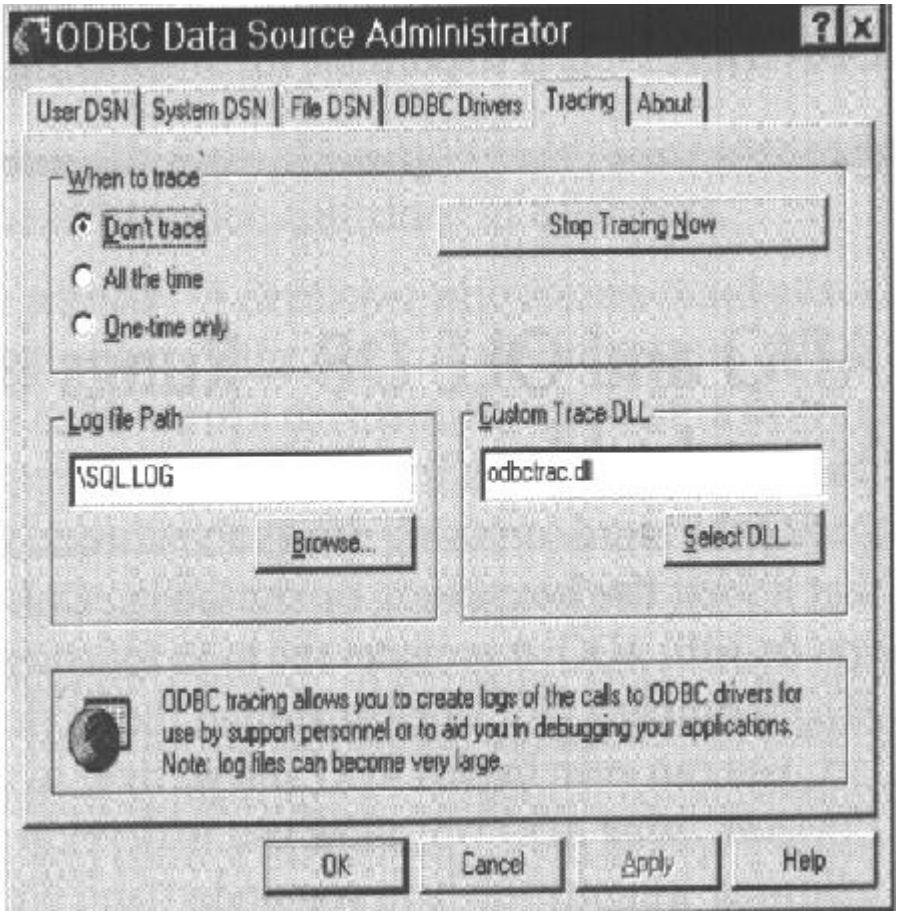
图 4.1　SAMPLE.DSN **文件包含了从** Visual C++**内部使用我的数据库时所需的全部设置**

记录 ODBC 事务

调试应用程序时加以记录，这一点很重要。ODBC Data Source Administrator（ODBC **数据源管理员）对话框也提供了这样的功能。可以选择跟踪你通过** ODBC **对数据库进行的各种事务。当然，这些记录可能会变得相当大，但我们并不是总要使用它们。**

开始记录事务时，只要双击控制面板(Control Panel）**中的** 32 **位** ODBC **小程**

序，打开 ODBC Data Source Administrator（ODBC 数据源管理员）对话框。选择 Tracing（跟踪）选项卡，你会看到如下图所示的对话框（请注意，该对话框的 Windows 98 版本略有不同）。



如你所见，有三个单选按钮确定何时跟踪 ODBC 调用。缺省设置是 Don't Trace（不跟踪）。如果打算调试单个应用程序，应该选择 All the Time（全部时间）。One-Time Only（只一次）在下一次连接期间跟踪 ODBC 调用——只要连接一拆除，跟踪就关闭。当用户带着特定的问题调用时，这是一个不错的选择。可以监视一次会话期间的连接，然后用该信息帮助创建排除错误的计划。

跟踪不会自动开始。还要单击对话框右侧的 Start Tracing Now（现在开始跟踪）。只要跟踪一开始，按钮标题就变成 Stop Tracing Now（现在停止跟踪）。再次单击按钮关闭跟踪过程。

唯一要担心的其它设置是 Log File Path（记录文件路径）。ODBC 通常在根目录下的 SQL.LOG 文件中放入事务信息。但是，也可以将该信息放在网络驱动器上，或者用户看不到的地方。在调试过程中，缺省位置通常就比较适宜。

注释 除非你想创建自己的日志记录 DLL，否则不要更改 Custom Trace DLL（自定义跟踪 DLL）域内的设置。在这里列出的 DLL——ODBCTRAC.DLL——负责维持事务记录。

4.2 ADO 和 OLE-DB——同一问题不同层次的解决方案

使用 ADO 时一件让人困惑的事是，需要理解它不是数据库访问的底层解决方案。OLE-DB 是使用 ADO 的基础；它也提供与数据库通信的基础。ADO 只是 OLE-DB 提供的服务的漂亮包装。事实上，如果你愿意，甚至可以绕过 ADO 而直接使用 OLE-DB。但是，使用 ADO 开发应用程序的速度要快得多。下面的章节会帮助你理解 OLE-DB 和 ADO。

理解 OLE-DB

那么，OLE-DB 是什么呢？顾名思义，它用 OLE（或者更具体一点，是部

件对象模型——COM）为数据访问提供一套接口。就像任何其它的 COM 对象一样，可以查询、创建和撤销一个 OLE-DB 对象。OLE-DB 对象的来源被称为提供者。你会得到各种 OLE-DB 提供者，它们是 Visual C++软件包的一部分，也有可能在厂商升级其数据库产品时提供。OLE-DB 的好处是，同一个提供者可以和任何 Visual Studio 产品协作：Visual C++，Visual Basic，Visual InterDev 和 Visual J++。

OLE-DB 还依赖于事件，这和任何 COM 对象一样。这些事件说明什么时候要求更新数据库表，以显示其它用户生成的新项，或者你所请求的表什么时候准备好供查看。还会看到这样的事件，它们用来发送信号通知各种数据库错误和其它要求马上轮询的活动。

Microsoft 定义了 OLE-DB 用户的四个主要类型。下面将各组予以分解，从总体上说明它们是如何帮助使用 OLE-DB 的。

- 数据提供者　用 OLE-DB SDK（软件开发工具箱）创建 OLE-DB 提供者的人。提供者用户与数据库和事件交互作用，发信号通知特殊事情的发生。
- 数据消费者　要求访问包含在数据库中的信息的应用程序、系统驱动程序或用户。
- 数据服务提供者　创建用以增强用户或管理员使用或管理数据库内容能力的独立实用程序（服务）的开发者。例如，开发者可以创建查询引擎，允许用户用自然语言请求数据库中的信息。服务和 OLE-DB 提供者互相协作，成为一个整体。

- 商务部件开发者　创建应用程序模块或部件，以减少创建数据库应用程序所需的编码量的开发者。部件可能和网格控件一样通用，可以快速地显示数据库中记录的子集，或者显示正被访问的数据库类型的专用信息。

那么，OLE-DB 和 ODBC 有什么不同呢？表 4.1 说明了这两个产品间的主要差别。我们会在本章的“应该在什么时候使用 OLE-DB、ADO 或 ODBC？”一节中讨论这些差别是如何影响你的使用决定的。

不要认为 OLE-DB 和 ODBC 是两种完全分离、可以互相替换的技术。ODBC OLE-DB 提供者允许你通过 OLE-DB 或 ADO 访问 ODBC 提供的所有功能。换言之，这两种技术是互补的，不能完全互相替换。可以用 ADO 或 OLE-DB 替换 ODBC 吗？可以，但如果这样做了，将得不到最佳的应用程序性能。OLE-DB 的整体作用是，拓宽可以用 Visual C++应用程序访问的数据库类型的范围。很明显，如果确实要用单个应用程序访问 ODBC 和列表数据，那么 OLE-DB 为此提供了较好的解决办法。

理解 ADO

现在你对 OLE-DB 比较熟悉了，那么 ADO 适合在哪里使用呢？如前所述，ADO 为访问 OLE-DB 提供者的功能提供了一种容易的方法。换言之，ADO 允许你快速地创建应用程序，允许 Visual C++处理你在直接使用 OLE-DB 时通常应考虑的一些细节。我已经在第一章关于 ADO 数据连接的章节中提供了关于 ADO 特征的概述，这里就不再讨论了。

表 4.1　OLE-DB 与 ODBC 的技术比较

元素	OLE-DE	ODBC	描述
Access 类型	部件	直接	OLE-DB 提供与数据交互作用的接口，用户访问数据要通过被设计成与 OLE-DB 交互作用的部件来完成
数据访问特殊化	任何列表数据	SQL	ODBC 总被设计成把 SQL 用作数据事务的基础。在一些情况下，这意味着程序员必须做出让步，强制数据符合 SQL 标准
驱动程序访问方法	部件	本机	如前所述，所有对 OLE-DB 提供者的访问都要通过使用各种类型部件的 COM 接口。ODBC 通常要求某类直接程序设计，且特别依赖数据库厂商所提供的 SQL 兼容性等级
程序设计模型	COM	C	OLE-DB 依赖于 COM 向程序员提供的对提供者的访问。这意味着 OLE-DB 是语言无关的，而 ODBC 则使用专用语言
技术标准	COM	SQL	OLE-DB 坚持 Microsoft 的 COM 标准，这意味着厂商和专用平台要比 ODBC 所使用的 SQL 技术标准多得多

与 OLE-DB 一样，ADO 也是基于 COM 的。它提供了一个双重接口：对本地操作是 ADODB 的程序 ID，对远程操作是 ADOR 的程序 ID。虽然注册表像

使用分室线程模型那样显示 ADO 线程，但 ADO 库本身是种自由线程。ADO 的线程安全依赖于你所使用的 OLE-DB 提供者。换言之，如果使用 Microsoft 的 ODBC OLE-DB 提供者，不会有任何问题。如果使用第三方的 OLE-DB 提供者，要在假定 ADO 线程安全之前检查厂商的文档（这是在 Internet 或企业网连接上使用 ADO 时的要求）。

使用 ADO 时会用到七个不同的对象。表 4.2 列出了这些对象，并描述了它们的使用方法。虽然 ADO 对象的功能层次要比先前技术提供的层次高得多，但其中多数对象类型是 Microsoft 已引进的其它技术的复制品。我们会在本章的概述一节更为详细地讨论 ADO 的类。

注释 一些 ADO 对象由接口而不是由实际类来表示。表 4.2 还说明了对象的关联关系，它帮助你理解如何导出并非由 Visual C++类直接表示的对象。

表 4.2 ADO 对象概述

对象	类	描述
Command	CADOCommand	命令对象使用连接或记录集对象执行任务。即使可以把执行命令作为连接或记录集对象的一部分，命令对象也要灵活得多，而且允许定义输出参数
Connection	CADOConnection	定义与 OLE-DB 提供者的连接。可以用这个对象来执行任务，如开始提交和回滚事务。它还提供了打开或关闭连接以及执行命令的方法

续表

Error		错误对象作为连接对象的一部分而创建。它提供关于 OLE-DB 提供者引起的错误的额外信息。单个错误对象可以包含多个错误的信息。每个错误对象都与特定的事件相关，如提交事务
Field		字段对象包含记录集对象内的单列数据。换言之，字段可以被视为表中的一列，它包含了某类与记录集相关的所有记录的数据
Parameter	CADOParameter	为命令对象定义单个参数。参数修改存储过程或查询的结果。参数对象提供输入、输出，或同时提供输入输出
Property		某些 OLE-DB 提供者需要扩充标准的 ADO 对象。属性对象代表了完成这一任务的一条途径。属性对象包含属性、名称、类型以及值信息
Recordset	CADORecordset	它包含了返回表中对所选独立元素查询和光标的返回结果。Visual C++让你选择用单个记录集对象创建连接和记录集，或者用现有的连接对象支持多个记录集对象

4.3 什么时候应该使用 OLE-DB，ADO 或 ODBC

要在访问数据时判断出应该使用哪一种技术，这并不容易。可能需要公用

实用程序来处理多个数据库类型；部分数据可能出现在本地硬盘驱动器上，部分在网络上，还有一部分在主机上。甚至客户安装在设备上的产品也会使这种选择更加困难。例如，你所期待的 ODBC 支持级别也许依赖于所安装的 Microsoft Office 的版本，因为这个产品不提供 ODBC 支持。你还会发现，ADO 类提供的对象和方法要比 ODBC 类多。ADO 可以提供程序中绝对必须具有的一些特性——例如，你会发现 OLE-DB 和 ADO 两者都支持 DFX_Currency，但在 ODBC 中没有对应的功能——但你要想掌握它们也必须付出一定的努力。

注 使用 ADO 而不是 ODBC 的一个原因是，ADO 提供的对象和方法比 ODBC 多。

选择 OLE-DB 或 ODBC 时，有几条一般的规则。因为 ADO 实际上只是 OLE-DB 的包装，所以这些规则也适用于它。下面提供一些基本的原则，可以用来帮助你决定选择 OLE-DB 还是 ODBC。

- 非 OLE 环境 如果要访问支持 ODBC 的数据库，而该数据库又在不支持 OLE 的服务器上，那么 ODBC 是最好的选择。
- 非 SQL 环境 ODBC 在处理 SQL 时非常出众。处理非 SQL 数据库时，OLE-DB 则具有非常明显的优势。
- OLE 环境 对支持 OLE 的服务器来说，选择 OLE-DB 还是 ODBC 也许是希望各半。如果有 ODBC 驱动程序可供利用，那么使用 ODBC 是一个好主意；否则，就只有选择 OLE-DB 了。
- 所需的互操作性 如果需要可互操作的数据库部件，那么只有选择 OLE-DB。

其它的问题往往是复合问题，或者至少是失去了可以用来区别这两种技术

的方便的规则。例如，有一些特征是 ADO 和 ODBC 所共有的。其中一个特征是，Visual C++允许你直接访问任一种技术。这意味着，可以完全访问 ADO 和 ODBC 都提供的每一个特征（是的，这确实不错，但也有缺陷，因为谈到访问，不能说一种技术明显地优于另一种技术）。

如果做了错误的决定，一些熟悉的技术实际上会帮助你把应用程序从 ODBC 转移到 ADO，或反方向转移。两种技术都依靠数据库对象来管理基础的 DBMS，而记录集对象则包含针对 DBMS 的查询结果。另外，ODBC 和 ADO 都以相似的成员来使用数据库和记录集对象。即使要对类和成员名称做一些更改，你仍会发现，ODBC 和 ADO 程序设计的代码非常相似。

ODBC 和 ADO 都依靠外部驱动程序来完成任务。但是，用 ADO 还可以从 Microsoft Jet 引擎获得额外的帮助。这个额外的支持是如何使程序设计更加容易的呢？考虑一下，要在哪里执行两个依赖不同引擎的数据库的外部连接。在 ODBC 下，必须人工执行全部所需的设置。使用 ADO 意味着，Microsoft Jet 引擎会为你执行这些设置。可惜的是，让 ADO 做这个工作意味着会失去控制级别和一些灵活性，而这正是你从一开始就想用 C++获取数据的原因。

注 ODBC 和 ADO 具有相似的特征，如记录集和数据库对象，尽管 MFC 利用这些对象的方式完全不同。

有一个地方绝对不能使用 ADO。如果要访问 16 位数据，就不能用 ADO。不管你怎么想，都必须用 ODBC。然而，几乎没有人还在使用 16 位数据库了。大多数的新工程都会使用 32 位接口，这意味着会有一个选择。老的工程已经有了嵌入代码的数据访问方法，所以你也实在没有必要在这里做决定。

ODBC 的一点不足是，不能像以前使用 ADO 那样跟踪事务。以 Microsoft Jet 引擎使用 ADO 时，应该得到事务的工作区级的支持。ODBC 只提供数据库级的事务支持，这就是说，可以从几个不同的工作区跟踪事务（这使得调试非常困难，还可能导致其它类型的问题）。

注释 处理 ODBC 数据库时，ADO 提供数据库级的支持；这和 ODBC 提供的支持层次是相同的。除非你使用 Microsoft Jet 引擎，否则得不到工作区级事务支持的好处。

4.4 Visual C++ ODBC，OLE-DB 及 ADO 类概述

用 C++创建数据库应用程序的讨论终于涉及到了类，为了进行实际的工作，你需要了解类。每个类都执行非常特殊的任务，所以，知道在哪里使用哪一个类至关重要。所有类的总目标都是从服务器获得特定的数据，并在屏幕或打印机上显示出来。

注 这确实可以帮助你把数据库对象看作是将数据从数据源装入你的计算机的一种管道。

我们要了解的第一个类是 Cdatabase(ODBC)或 CADOConnection(ADO)。用这些类创建的对象让你得到对数据库内数据的访问。既可以创建指向特定记录的指针，也可以下载整个查询；数据库对象创建你需要的连接。获得的数据访问的类型依赖于是创建了窗体视图（所有数据的单记录显示）还是记录视图（与

特定标准相匹配的所有记录的网格显示）。

一旦连接了数据库，就要有某种容器来保存它所包含的内容。CRecordset(ODBC)和 CADORecordset(ADO)类在这里发挥作用。记录集保存最终要在屏幕上显示的数据。这有助于把记录集看作是保存数据库中数据的容器。如下所述，有三种类型的记录集：

- 表　表型记录集代表了数据库某个表内的数据。你可以对这样的单个表进行任何处理，包括添加、删除或编辑记录。
- 动态集　需要用查询从数据库的一个或多个表中提取信息时，会用到动态集型记录集。与表型记录集一样，可以添加、删除或编辑动态集型记录集中的记录。
- 快照　这是数据库一个或多个表中所包含的数据的静态拷贝。与动态集一样，要通过查询来提取信息。与动态集不一样的是，你不能修改快照中记录的内容。但是，可以用快照型记录集查找数据或产生报告。

你可能已经注意到了，我在谈论记录集时提到了术语“查询”。查询就是一个简单的提问。查询时所做的一切工作就是要求数据库提供一批符合特定标准的记录。使用 ODBC 时，要依靠 CRecordset 类数据成员来更改想要看的记录的查询。ADO 有一点不同。可以用 CADOCommand 类创建特殊的查询，或用 CADOConnection 执行标准的查询。CADOCommand 要比 CADOConnection 灵活得多。可以用它来执行多种和数据库有关的任务。例如，可以用它操纵数据库的结构或者执行其它的管理任务。

本章中我们要考察的最后一组类是实际浏览数据时使用的类。

CRecordView(ODBC)**和** CADORecordView(ADO)**类允许你实际查看检索到的数据。本质上说，这些类要完成的任务就是将数据从记录集对象移动到对话框或窗口上的控件中。你还可以使用这些类来检测何时到达记录集的开始或结尾。**

第5章　数据库创建概述

绝大多数的应用程序由于某种实在的需求而开始设计。假定你的硬盘需要某种特殊的维护。下一步就是描述你所需要的功能。例如，一个实用程序可能需要访问磁盘资源，确定是否有问题，并最终解决所发现的问题。一考虑到某种功能，就可以开始查看用户接口，最后创建使设备正常运转所需的代码。

所有这些步骤，数据库应用程序都要求。你依然要有已知的应用程序需求，然后决定那个应用程序要做什么。不过，数据库应用程序还要增加一步。数据库应用程序的主要功能是访问数据，并有条件地允许用户操纵它（当然，这是一个非常简化了的说法，实际过程要复杂得多）。在真正进行数据库应用程序设计之前——明确地说，在创建任何用户界面元素之前——必须首先设计数据库本身。

决定如何设计你的数据库的部分过程是规划出你计划如何访问数据库以及计划在数据库中放入多少数据。例如，如果唯一的目标只是存储一些个人姓名和地址，那么就不会需要基于服务器的 DBMS（数据库管理系统）。你还要清晰地知道所用的程序设计语言是否能够操作所使用的 DBMS。在第 4 章中，我们研究了三种从 Visual C++访问数据库的方法：ODBC，OLE-DB 和 ADO。我甚至说明了如何设置 ODBC 的数据源名称(DSN)。最后，我们讨论了不用获得

额外的驱动程序即能访问的 DBMS 的类型，以及在需要时获取驱动程序的途径。不过，知道如何访问数据库并不是说你已经知道数据库都包含些什么。

注 在可以设计数据库之前，要知道如何访问它以及数据库的内部结构的工作方式。

本章中，我们将学习数据库的内部知识——并探查与 DBMS 相关的数据。我们将简单地介绍一下数据库的基础知识。例如，学完本章后，你会知道什么是表以及如何定义表。我们还要学习查询和索引，还要考察几个界面元素，比如窗体等，但这只是最基本的知识。最重要的部分是，你会理解数据库是如何运行的，这是用 C++访问它们的先决条件。

由于篇幅所限，还有一些问题本章中没有讨论。我不想花很多时间去探讨关系数据库理论——市面上这样的书很全。本章结束时，你可能不会全面地掌握 SQL（结构化查询语言）或完全理解用户界面设计。总而言之，通过本章，你会掌握创建后续章节中用到的简单数据库所需的一切知识——应该把本章看作是数据库的入门资料，仅此而已。

Web 链接 如果需要更多的关于特定数据库问题的信息，而不是仅限于大多数书籍所提供的或某个人所谈到的信息，可以试一试新闻组。例如，你会在 comp.theory.info-retrieval newsgroup 中找到很多理论信息。一系列完整的 comp.database.<product name>新闻组，比如 comp.database.ms-access，也提供了许多帮助。你会看到，这些新闻组可以帮助你发现隐藏在数据库设计之后的理论问题。它们还可以指导你克服设计自己的数据库时所面临的障碍。

5.1 构造块概述

每个数据库都是由模块化的片组成的。有存储数据的片、组织数据的片，还有的片允许你查找数据库中特定片的信息。数据库还包含接口元素和允许你操纵所包含数据的代码片。然而，在设计 C++应用程序时，你多半会忽略后两种模块化的片。事实上，我们在本章中不讨论编码，因为编写 C++应用程序不需要它（我们只简单地讨论窗体，因为它们在你从 Visual C++访问数据库结构之前，提供了测试 DBMS 中你的数据库结构的便利方式）。

注释 我们要为本章中所有的示例使用 Microsoft Access。并不是因为 Access 更好一些，而是因为它提供了大量的工具，可以使数据库设计过程更加容易。Access 还是 Microsoft Office 专业版软件包的一部分，所以在阅读本书时，很可能你们中许多人的手头都有 Access 的拷贝。请记住，你应该会有所收获，并把它们应用到其它的 DBMS 产品中去。

图 5.1 概述了 Visual C++、数据库管理器以及数据库是如何交互作用的。如果你现在实在不理解图中的一切是如何进行的，那也无须担心。我们将在随后的章节中讨论所有的元素。现在，这张图只是大体说明各种部件是如何连接到一起的。

可移植性 开始设计过程之前，一定要确保数据库和你选择的 DBMS 实际上可以和 Visual C++相协作。使用 ODBC 之外的 ADO，OLE-DB 和 DAO 的能力应该能够极大地增加找到正确解决办法的机会。另外，如果决定从多个平台访问数据库的话，需要保证有办法以想为其编写应用程序的所有平台上使用数据库，并保证 DBMS 支持文件和记录锁定的标准方法。简言之，在最后决定使用哪一个 DBMS 之前，必须检查每一个可计算性问题，尤其是计划在多种机器上使用 DBMS 的情况下更应如此。

数据库

最好把数据库看成是一个容器（见图 5.1）。数据库用来存储要用于存储和操纵数据的全部元素。大多数的数据库是单个的文件，比如 Access 用的 MDB 文件。也可以使用有分离文件的产品，文件都存储在一个像目录的地方。在这种情况下，目录就像数据库——它是把所有其它元素保持在一起的容器。

注释：Ashton-Tate（dBASE 的发明者）首次在单个 Xbase 表格（DBF 文件）中使用了“数据库”这一术语。事实上，这个文件的扩展名 DBF 的意思就是“数据库文件”。这一销售策略带来的后果是，一些人仍然对什么是表以及什么是数据库感到迷惑。数据库是一个把所有其它数据元素保存在一起的容器，其中包括代码、窗体、表、查询、索引和报告。不管你现在在想什么，遍及全书的术语“数据库”总是指这样一个容器，它将一个数据集的所有数据元素都保存在一起。

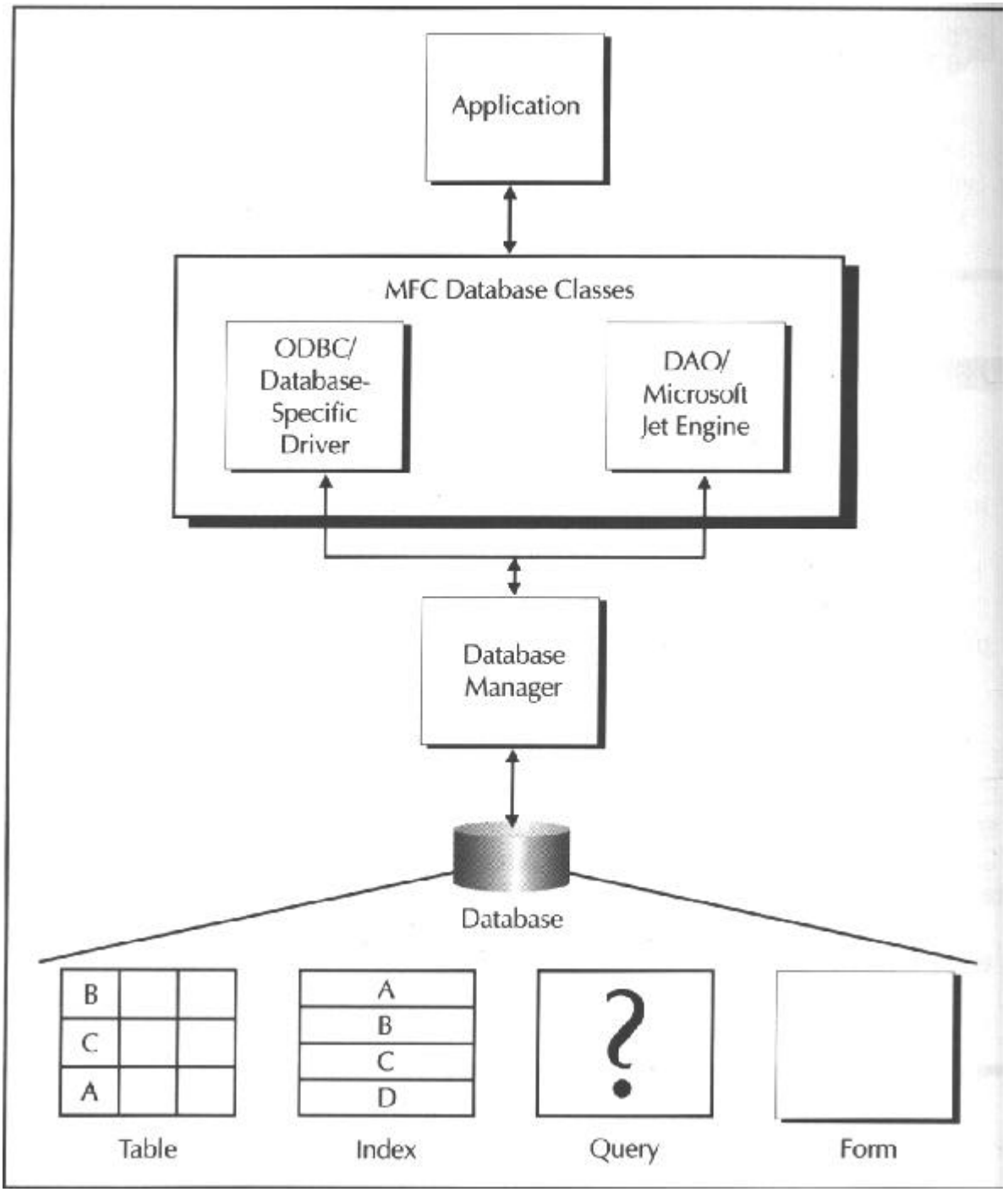


图 5.1　Visual C++、**数据库管理器和数据库本身之间的关系**

数据库并不只是端坐在那里，看上去很可爱。一个被称为数据库管理器的特殊应用程序可以在文件内部查看并提取所需的一切。如果你正在建立数据库，可能想看看已经定义的查询或者窗体。另一方面，作为一个用户，你唯一的兴趣可能是包含在表内的一些难以理解的比特。不管你需要什么，数据库管理器正是这样的应用程序，它在数据库内查找并为你提取所需要的东西。本质上讲，数据库管理器像一个正在碗柜这个容器中找东西的人那样，寻找某个玻璃器具或者什么吃的东西。

注　和任何用户一样，Visual C++请求数据管理器从数据库获取它所需要的东西。

把数据库管理器和数据库放在一起，就形成了数据库管理系统(DBMS)。在Visual C++的情况下，用来处理数据的数据库管理器和你编写的程序共用数据管理器的规则。换言之，用户要提出请求，必须和应用程序交互作用。反过来，应用程序请求数据管理器提供用户所请求的数据。数据管理器会返回数据，应用程序将其显示出来。这听起来好像有很多工作要做，实际上并非如此。对 C++ 程序员来说，大部分工作是自动完成的（至少用 MFC AppWizard 创建应用程序时是这样）。

表

表是数据库内唯一可以保存数据的结构。其它每种结构都被设计成以某种方式处理数据：窗体用于显示数据、索引用于排序数据、查询寻找特定的数据片。实际存储数据的唯一结构就是表。

那么，表是如何拼装起来的呢？图 5.1 已很好地说明了这一点。请回想一下你用笔和纸创建的其它表（或者是你在类似本书的书籍中看见过的表）。表通常包含行和列。同样，数据库中的表也包含行和列。

行就是记录。每个记录代表了要收集的所有数据的一次出现。例如，如果你在创建姓名和地址表，那么行（记录）应该包含一个人的所有信息（至少应该包含设计表时要求保存的信息——我们会在后面看到，表是以节省数据库服务器空间的方式组合起来的，你仍然可以看到正在过目的任何数据的完整图片）。

注　行（记录）总是包含要存储在数据库中的数据实例。

每一个表列（或称做字段）都包含一类数据。例如，可能有名为 Last_Name 的表列（字段）。这个字段应包含你的姓名与地址数据库中所有人的姓名。字段只保存一类数据。你不能用单个字段在一个记录中保存地址信息，而在另一个记录中保存年龄信息。如果既要保持地址信息又要保持年龄信息，就必须构造两个字段。

注　列（字段）包含一类数据，如姓氏或者地址。

从表中获取特定的一段数据或者数据元素很容易。你所需要完成的任务就是请求特定的行（记录）和列（字段）。从数据库中获取数据的过程——不管它是包括一个数据元素还是包括表提供的所有记录——都称为查询。本质上讲，在你向数据库管理器提出一个问题后，它就提供与该问题相匹配的数据予以响应。Visual C++在提出问题时，很大程度上要依赖一种特殊的语言，这就是 SQL（结构化查询语言）。但是你会发现，在使用 ADO，OLE-DB 或 DAO 时，也

可以利用其它的机制。随着课程的进展，我们会越来越熟悉 SQL，但是如果想设计复杂工程的话，就要花一些额外的时间来学习这门语言。

注 创建查询就像是询问数据库关于表内容的问题。

索引

如果再看一下图 5.1，你就会注意到，表第一列中的项并没有按字母顺序排序。没错，数据库不存储你以特别的顺序发送给表的项——事实上，如果数据库不排序的话，其效率可能会更高。如果每次某个人添加或删除记录时，数据库管理器都要重新排序表中所有的记录，你可能就做不了太多的工作了。不用说，你并非总是需要数据的顺序相同，所以两个人没有必要去争论数据的顺序是什么。一个人可能想要地址表以姓氏来排序，另一个人可能又需要表按 ZIP 代码排序。

注 数据库把数据放在表的末尾，不能保证会以什么样的顺序来查找未编索引的表中的数据。

那么，如何以某种顺序获得所需的信息呢？答案是索引。图 5.1 说明了每个记录都要考虑的两个原则。第一，索引不能包含整个表的所有项——它只包含一个子集。在这里，它只包含第一个字段（列）内的项。第二，索引是有序的。索引包含了指向表中的记录的指针，不过图 5.1 没有显示出来。换言之，在选择记录时所看到的数据，和数据库管理器用来查找整个记录的指针是成对的。排序索引允许你快速地查找想要的那类数据。指针允许你查找表中的整个记录。使用索引还允许两个人以两种不同的顺序访问一个表中的数据。因为索引有序

而表无序，所以可以为用户所需的每一种数据排序方式定义一个索引。

没必要把索引限制在单个表内使用。很多数据库管理器允许你创建跨越多个表的索引，只要那些表是相关的即可。例如，为了掌握客户的采购情况，可以创建两个表。第一个表包含客户列表，而第二个表包含每次采购的细节。索引可以用消费者 ID 字段加采购编号来排序两个表，首先是以客户为序，然后是以采购编号为序。

注　只要表以某种方式相关，索引就可以跨越多个表。

我们来看一看这个多表索引概念实际上是如何工作的。图 5.2 说明了多表索引的一个示例。请注意，Clients（客户）和 Purchases（采购）表间是有关系的。两表中编号为 0001 的 Client ID 识别名叫 Ann 的客户。但是，Clients（客户）表只有 Client ID 的一个拷贝，因为一次只有一个客户可以使用 Client ID。因为客户可以多次采购，所以 Purchases（采购）表包含每个 Client ID 的多个拷贝。请注意，在 Purchases（采购）表中记载 Ann 采购了$20。这次采购由编号为 97001 的 Purchase ID 识别。此外，每一个 Purchase ID 都是唯一的；否则，无法将一次采购与另一次采购区分开来。现在看一下索引。它包含两个值，Client ID 和 Purchase ID。它先由 Client ID 排序，然后由 Purchase ID 排序。虽然表是无序的，但是该索引却有序。索引中的指针允许你在两个表中查找信息。

不要认为索引也被限制在表内的直接项上。很多数据库程序员都发明了组合表项的方法，甚至可以用特殊等式产生非表的索引数据。很明显，数据库管理器制定的规则限制了把什么放入索引的选择。换言之，如果数据库管理器不允许，就不能把某种数字计算添加到索引中去。同时，如果以最初预想的顺序

在屏幕上显示数据时有麻烦，可以先做一下试验。

注 一些 DBMS 不限制你在索引中使用哪些字段名；你也可以使用复杂的表达式。

可惜的是，Access 没有提供在数据库设计视图中创建复杂索引的简明方式。可以创建包含多个字段的索引，可以分别控制那些字段排序的顺序，但是不能直接使用等式或者其它复杂的指令。在一些情况下，可以用 SQL 语句创建复杂的索引，但其本身就很复杂。幸运地是，你可以把复杂的索引作为查询的一部分来提供，我们会在本章的后面讨论这一点。

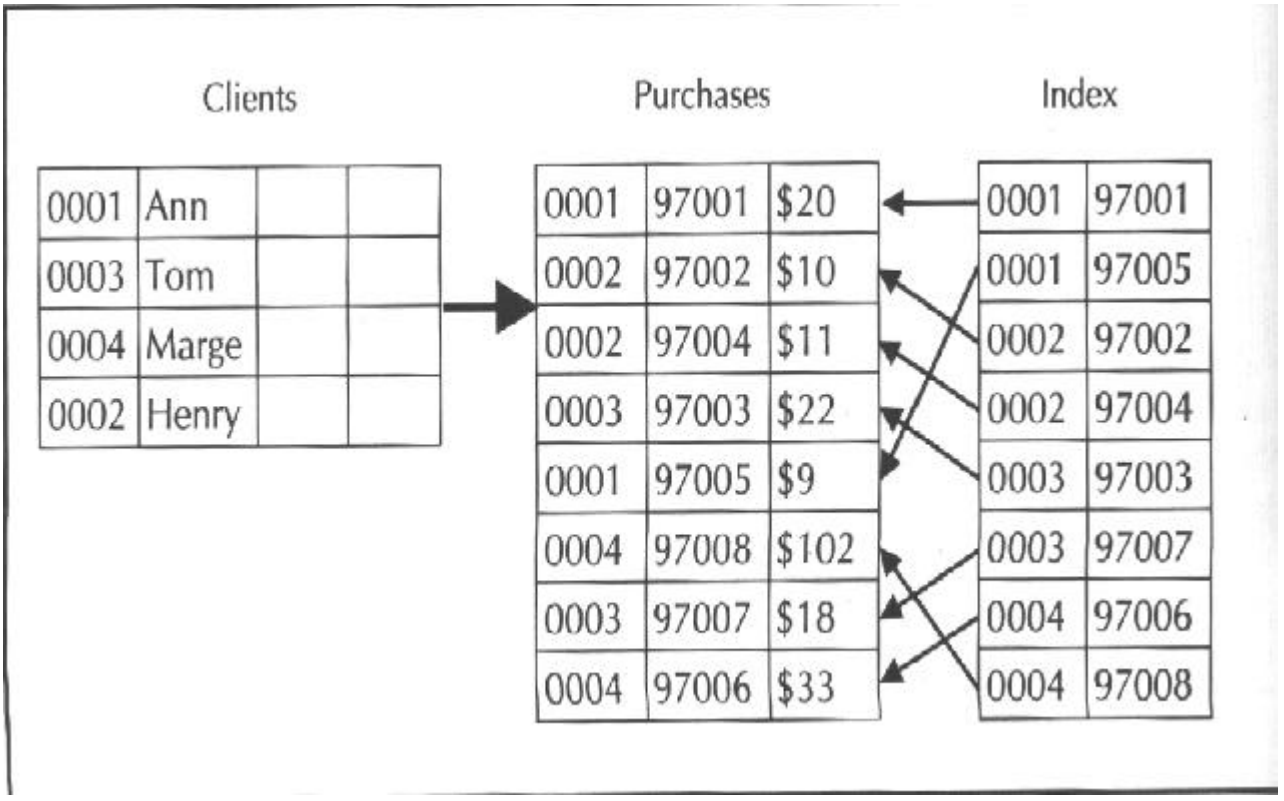


图 5.2 索引可以跨越多个表

技巧 DBMS 不允许直接向索引中添加等式或者计算值，但是，你可以通过向表中添加特殊的计算字段来实现这一点。该字段可以包含计算值，而索引又包含这个特殊的字段。例如，可以用 Access 完成这个工作。它允许用 Expression Builder（表达式生成器）给一个字段赋缺省值。如果用户从来没有见过这个字段，那么结果就是你可以为了编索引而使用的特殊字段。如你所见，结果是一样的，只不过完成的方法稍稍复杂了一点。

查询

再看一下图 5.1。我用来定义查询项的图标表示这是一个大概的看法——查询是向数据库提出的有关它所包含数据的提问。可能会有人向你展示难以理解的复杂的 SQL 语句，试图让你相信，查询并不是一个简单的过程。一个简单的事实是，查询就是一个提问。

注 始终要记住，查询就是向数据库提出的有关它所包含数据的提问。

每个查询都有特定的部件，但这个信息只是帮助你以 DBMS 可以理解的方式来简述这个提问（请记住，DBMS 实际上对操纵数据库的内容负责）。记住这一点，我们来看一看比较简单的 SQL 语句，它允许你查看两个相关表中的所有记录。

```
SELECT Foods.*, Orders.*
    FROM Foods INNER JOIN Orders ON Foods.Food_ID = Orders.Food_ID
```

```
WHERE ((("Food_ID")<>"VEG-00001-SPL"))
ORDER BY Foods.Food_ID;
```

首先我要指出，在使用 Access 时，没有必要使用像这样的 SQL 语句，除非你实在想用。Access 会用 Select Query（选择查询）对话框帮助你创建 SQL 语句。现在让我们看一看这条查询语句。

我们要做的第一件事是，选择两个表以及这两个表内的所有字段。SELECT 语句要 DBMS 选择一些数据。Foods（食品）和 Orders（订货）是两张表。出现在表名后的星号(*)要 DBMS 选择两张表中所有的字段。如果想选择特定的字段，应该以<table name>.<field name>格式输入字段名。

SQL 语句的下一行看起来确实很复杂，但它所要做的，只是要 DBMS 建立两张表的关系。FROM 关键字告诉 DBMS 主表是 Foods（食品）。INNER JOIN 说明如何创建与从表 Orders（订货）的关系。ON 关键字说明我们准备建立关系的字段。在这里，两张表都有一个同名的字段，我们就用它来创建关系。创建关系的标准是，有两个包含相同类型信息的字段，就像我在图 5.2 中说明的那样。在这种情况下，两个相关的字段都是 ClientID。对主表中的每一个项来说，从表通常都包含多个项，虽然这并不是绝对要求的（对主表中的每一个记录，从表不是非得有一个项与之对应）。

可以用 SQL 来限制实际获取的记录数。在这里，我们告诉 DBMS，不想看到 Food_ID 字段等于 VEG-00001-SPL 的任何记录。很明显，可以用任何标准来限制取回的记录数，选择标准可能变得非常复杂。

最后，ORDER BY 关键字告诉 DBMS 你想如何排序记录。在这里，我们想

基于 Food_ID 字段的内容以升序次序来排序记录。因为 SQL 允许完全利用表达式，所以你可以使 ORDER BY 子句要多复杂、就多复杂。可以用等式或其它任何所需的标准，以正好需要的顺序获取想要的数据。

Web 链接 如果你确实想熟练掌握 SQL，请看 http://www.opengroup.org./ot/search.htm 处的 Open Group Web 站点（Open Group 是 OSF 和 X/Open 的组合）。输入 SQL，然后在询问想要寻找的内容时单击 Search（搜索）按钮。你会发现一系列的 SQL 规范。

与 Access 不同，Visual C++在大多数情况下会要求你手工创建这些 SQL 语句。时间不长你就会发现，在 Access（或你正在使用的任何 DBMS）中创建所需要的查询是提高开发速度的一种方式，尤其在你对编写 SQL 语句了解不多的时候更是如此。

注 窗体在屏幕上表示用查询从数据库收集到的数据。

窗体和报告

到现在为止，我们谈论了很多管理和存储问题。不错，在可以使用数据库之前，它要求内部管理结构全都正确，但是，如果不能以某种方式使用收集到的信息，那么确实没有任何理由说工作已经完成了。窗体和报告是实际使用数据库信息的两种最普通的方法。

注 报告通过打印表示用查询从数据库收集到的数据。

通常，窗体表示在屏幕上看到的数据，而报告表示发送到打印机的相同数

据。窗体常常显示为已请求的所有记录的网格，或者一次只显示一条记录。报告总是显示已收集到的一系列完整的数据，但它通常不使用网格格式。典型情况下，报告提供已请求数据的某种结构化视图。

深入窗体

我们再来谈一谈窗体。如前所述，通常用两种方式来表现窗体中的数据：网格中的所有记录或者一次显示一条记录。图 5.3 显示了典型的网格视图，图 5.4 则显示了典型的单记录细节视图。网格法的主要优点是可以得到数据的整体印象。使用这种视图时，正确地排序数据一定会有好处——当记录数超出一屏信息时，可以容易地看到记录之间的关系。例如，在以网格显示之前，可以用产品类型来排序库存数据库，这样就可以看到库存中有多少相似项。

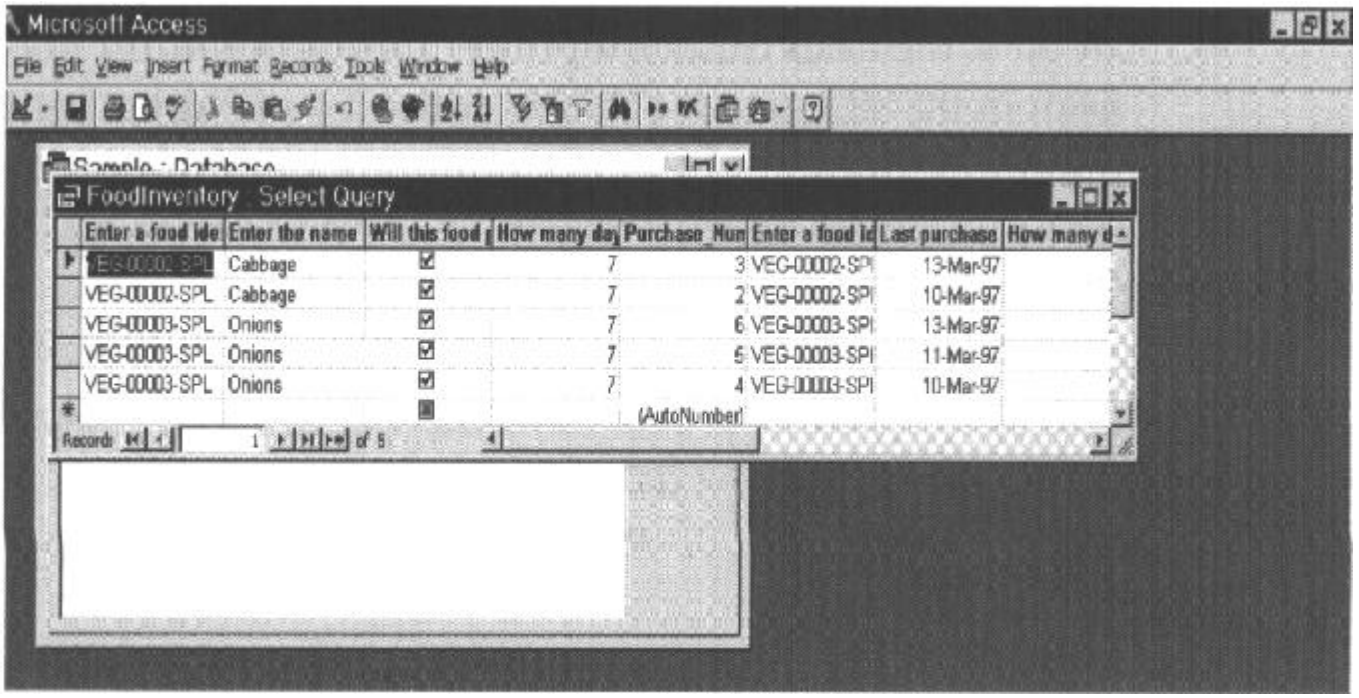


图 5.3　**网格视图允许你一次看到多条记录**

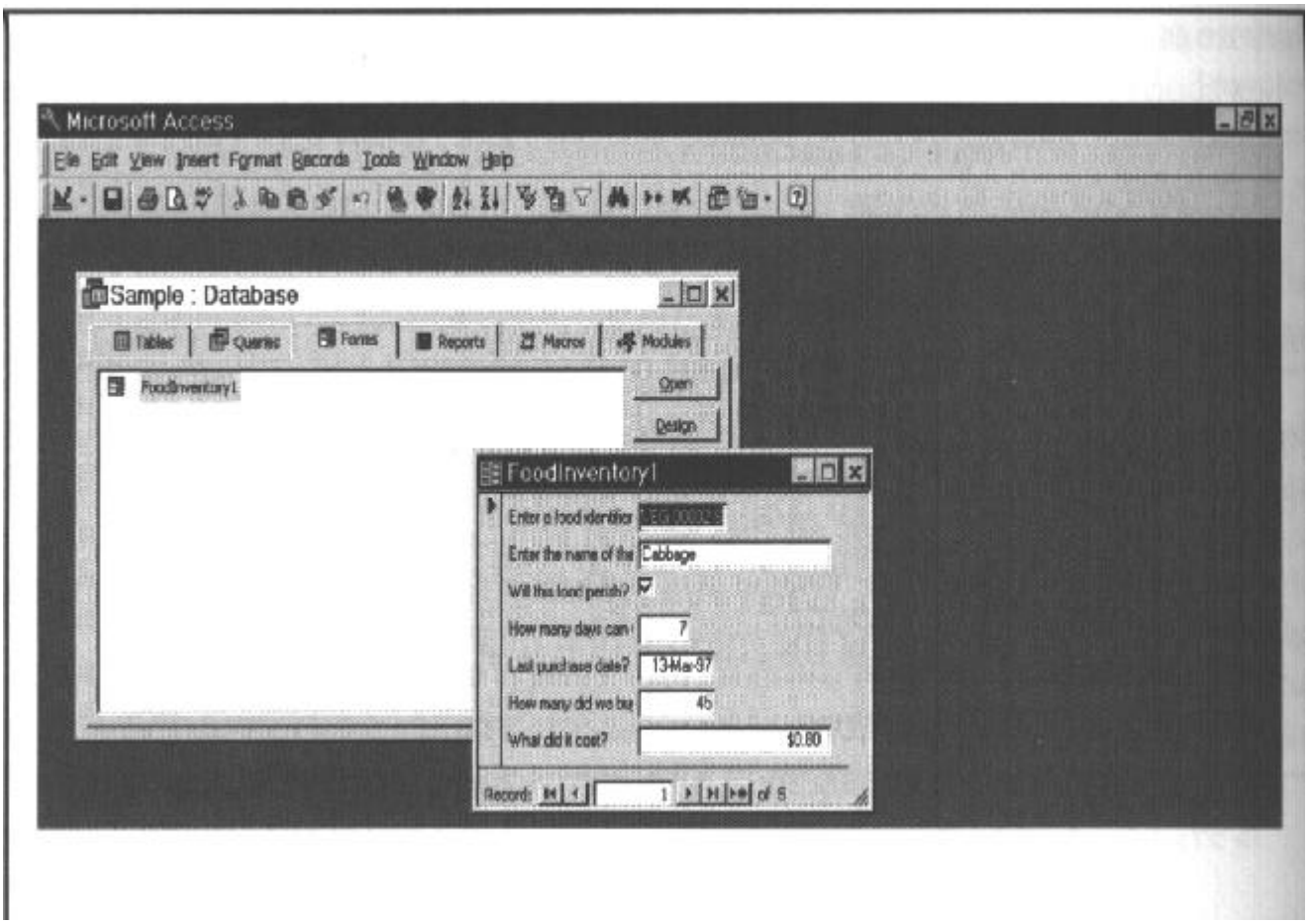


图 5.4　单记录细节视图允许你看到一个记录的所有信息

使用数据网格视图的缺点也十分明显。不但所有的字段都挤在一起，而且如果不水平滚动的话，很可能看不到表（或表集）中全部的字段。当然，可以很快地获得许多记录的概述，但是看不到个别记录的完整情况。一些数据库应用程序克服了这种限制，允许从网格视图的一组记录中选择单个记录，然后再转换到单记录细节视图方式。

技巧 一定要放置你认为用户在设计网格视图窗体时最需要的字段。这样，才能极大地减少用户查找重要数据时必须做的水平移动次数。一旦根据重要性对数据进行了排序后，就可以实现第二步，把相似的数据聚集到一起。例如，把所有的电话号码保存在一个地址数据库中是非常有用的。这样，试图查找特定人的住址时，只要逐个调用号码就可以了。可以通过事件监视用户的使用，或者用应用程序实际地观察它们，以此来测试网格的排列。用户查找数据时所必须做的移动越少越好。

单记录视图有精确表示一个记录的优点。你可以看到记录所提供的一切。当然，一看到图 5.4，这种表示的缺点立刻就明显了。不像网格视图，单记录视图不提供任何种类的概述。不能用它来分析趋势，或者得到整个表内容的更完整的情况。由于这个原因，单记录视图通常被保留作表数据的详细视图。一旦用网格视图找到了想要的东西，就可以用它来编辑或者验证要删除的记录。

注 用网格视图向用户提供可以用来选择记录的概述，然后用单记录视图让用户一眼就看到记录的全部细节。

第三种窗体视图你可能也会经常使用。它由添加了网格的单记录视图窗体组成。单个记录显示主表的内容。例如，它可能显示如姓名、地址和电话号码这样的客户信息。网格包含与主记录相匹配的从表内的记录。例如，网格可能包含一年中某客户进行的所有采购。

注 始终要避免数据过多而使得窗体杂乱无章。如果有必要的话，可使用多重窗体来避免屏幕杂乱。

可惜的是，要做出一个出色的组合型窗体是十分困难的。除非严格限制网

格显示的字段数，否则网格会占据很多的屏幕资源。结果，很多组合窗体看起来都很拥挤，而且就是因为有太多的东西要看，反而很难看得见数据了。问题是，你想使查找数据很容易，而这限制了用户在任一给定窗体上的位移量。

单记录视图的另一种窗体使用多重窗体来避免屏幕杂乱的问题。假定实际上需要 50 个字段来描述才能了解库存中的一个项目。视图在一个屏幕上显示那么多的字段，就会产生极大的杂乱，没有人可以找到所需要的东西。在这里，应该创建一个概述记录中的数据的主窗体，然后由辅窗体提供详细的资料。例如，可以在一个辅窗体上放入库存注释，因为注释往往要占据很多的空间。还可以在分离的窗体上放入特定项目的厂商数据，因为在很多情况下，一般来说你不会再排序新的项目了。当然，添加辅窗体就是向主窗体中添加细节按钮，这会进一步减少可用的屏幕空间量。要权衡屏幕杂乱和一个窗体上按钮过多的混乱因素。在一些情况下，最好的答案是，除非确实需要某条信息，否则不要显示它。

深入报告

最初，人们设想计算机可以产生无纸办公室。现在好像并没有实现这一点。用计算机产生的报告比任何用打字机的办公室产生的报告还要多。事实上，创建报告是很容易的，以至于还没有真正地去思考时，报告纸就已经开始产生了。

传统的报告赋予你这样的能力，将信息存储在数据库中，并以纸的形式输出。但是，报告不必要再出现在纸上了。例如，如果你有一个 Web 站点，需要使雇员及时了解公司当前的状况，就可以发表能在 Internet 浏览器中阅读的形式的报告。电子邮件也在通信中发挥了更大的作用；你可能会发现，报告必须是

电子格式，这样才能把它邮寄给别人。

不论报告采取哪一种形式，它都具有一些确定的特性，需要在创建过程中加以考虑，这些特性是：组织、目的和接受对象。下面各段就讨论要着重考虑的这些问题。

组织　在可以组织报告之前需要考虑，想在报告中看到哪种信息，该信息有多少是重复的。除去不需要的数据，为你想要查看的数据腾出空间——当使用已打印的媒体时，这种考虑非常重要，因为在使用数据的时候就不能再用上它了。腾出空间的另一种方式是，使所有重复数据都在报告的同一行上。这个过程称为分组。创建分组不但节省了空间，还可以帮助报告的阅读者更快地查找到所需的信息。

目的　报告的目的确定，应该包括多少信息、摘要是否会替代详细的资料清单。既要给读者提供所需的全部信息，又不能让无用的信息使报告显得杂乱，这一点很重要。例如，没有必要把客户数据库的每一个细节都告诉经理；他们真正感兴趣的是最后一行：每个客户采购了多少，或者是每次采购的金额有多大？

我们再来看一看我在图 5.2 中介绍过的 Clients 和 Purchases 表。如果你为此表创建了采购报告，那么肯定想包括 Purchases（采购）表中的全部摘要信息。但是，为了节省空间，最好把所有像 Client ID 和 Name 字段这样的客户信息用作标题，分组采购信息。在像这样的报告中，可能不需要所有的客户信息。例如，如果这个报告的目的是产生一个电话号码表，就可能无须添加地址信息。

如果想创建一个关于客户的一般报告该怎么办呢？在这种情况下，可能要

包括 Clients（客户）表中的所有客户信息，而排除 Purchases（采购）表中像 Purchase ID 这样的信息。另外，可能只要简单地合计每个客户的采购值，并在报告中列入单个数字，而不是创建每次采购的详细清单。

接受对象（观众） 观众会影响编制报告的类型。例如，你可能会发现，给同事看的报告可以采用非常简单易读的格式。你可能需要这样的格式，因为这样的报告包括了很多详细的信息。另一方面，从管理角度讲，又要使报告更精致一些才行。

技巧 不要低估你作为程序员对报告的需求。当然，可以试着用屏幕上的窗体排除所遇到的数据库设置问题。但是，报告可以帮助你查找隐藏更深的错误。例如，你可能会发现，数据库有微妙的数学错误，你只有在打印数据库内容时才会看到它。如果只是在屏幕上看数据库内容，这个错误可能不是很明显。一定要把你设计的报告实际打印出来，才能看到它们在显示数据库的内容方面同样非常有效。虽然你是在用 C++设计应用程序，但是了解 DBMS 本身能干些什么，会对在 Visual C++ 环境内设计更好的报告有所帮助。

5.2 创建数据库

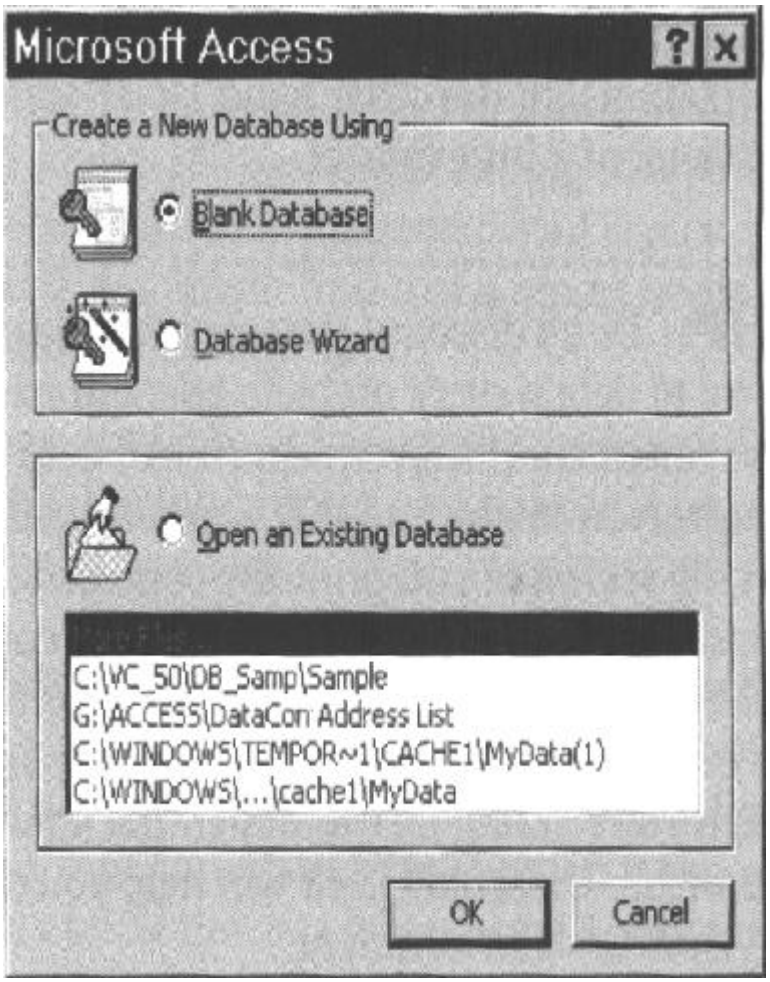
现在看一看数据库。这儿的数据库不会是你曾经见过的最复杂的数据库。实际上，开始时它只包括两个表。本节的整个目的就是让你从基础开始。虽然

我们要为这个示例使用 Access，你当然也可以把学到的基础知识应用到其它数据库产品中去。

注释 我们在全书中都要用到这个示例数据库，所以即使你不打算把 Access 用作你的 DBMS 理解它的创建过程也相当重要。后面数据库章节中的所有示例代码都是以这个示例表为基础的。你看到的演示原则上同样适用于其它 DBMS。请记住，Visual C++使用 ODBC 和 DAO 两种类，它们都不依赖于特定的数据库实现。

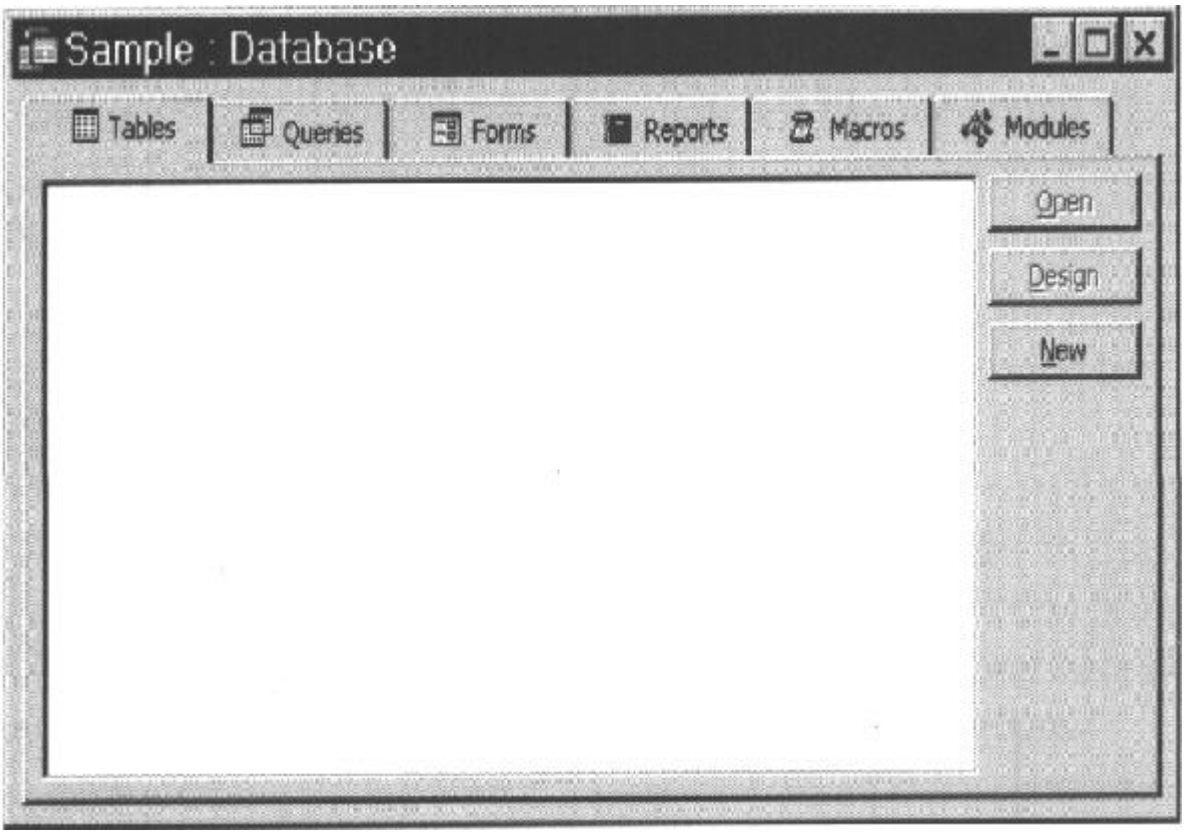
你需要完成的第一件工作是创建一个新的数据库。下面的过程帮助你从头开始。所有的示例程序都假定你将数据库命名为 Sample，并把它存储在 DB_Samp 目录中，当然任何名字和位置也都是可以的。

1. 启动 Microsoft Access。你会看到如下图所示的 Microsoft Access 对话框，它询问你想要做些什么。



2. **选择** Blank Database**（空白数据库）选项，然后单击** OK**。会看到** File New Database**（文件新数据库）对话框，它看起来与** File Save**（文件保存）对话框十分相似。**

3. **输入数据库名（本示例使用** Sample**），选择保存数据库的目录（本示例使用** DB_Samp**），然后单击** Create**（创建）。这时，会出现如下图所示的空白** Database **对话框。**



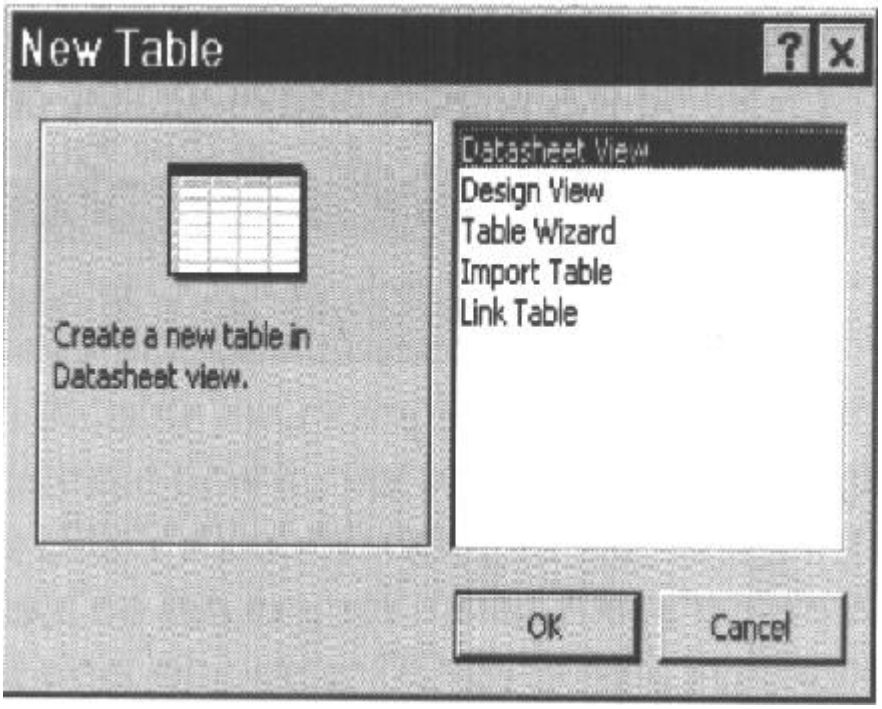
现在有了可以任意使用的空白数据库，可以开始用表、查询和数据库具有的其它部件来填充这个数据库了。下面的章节会说明，如何创建本书中示例程序所需的元素。当然，你也可以根据需要添加任意多的数据库元素。为进行实验而建立一个示例数据库是个不错的想法。

注释 下面章节中的所有过程都假定，你打开了示例数据库并准备使用。如果在前面的步骤中都遵守了示例的约定，应该会看到 Sample: Database 对话框。

5.3 添 加 表

在示例数据库中，我们需要使用两个表。第一个表 Foods（食品）包含一系列食品库存项。第二个表 Orders（订货）包含食品项的订货信息。我们从创建 Foods（食品）表开始。

1. 选择 Sample: Database（示例：数据库）对话框的 Tables（表）选项卡。
2. 单击 New（新建）。会看到如下图所示的 New Table（新建表）对话框。



3. 加亮 Design View（设计视图）选项，然后单击 OK。会看到如图 5.5 所示的 Table（表）设计对话框。

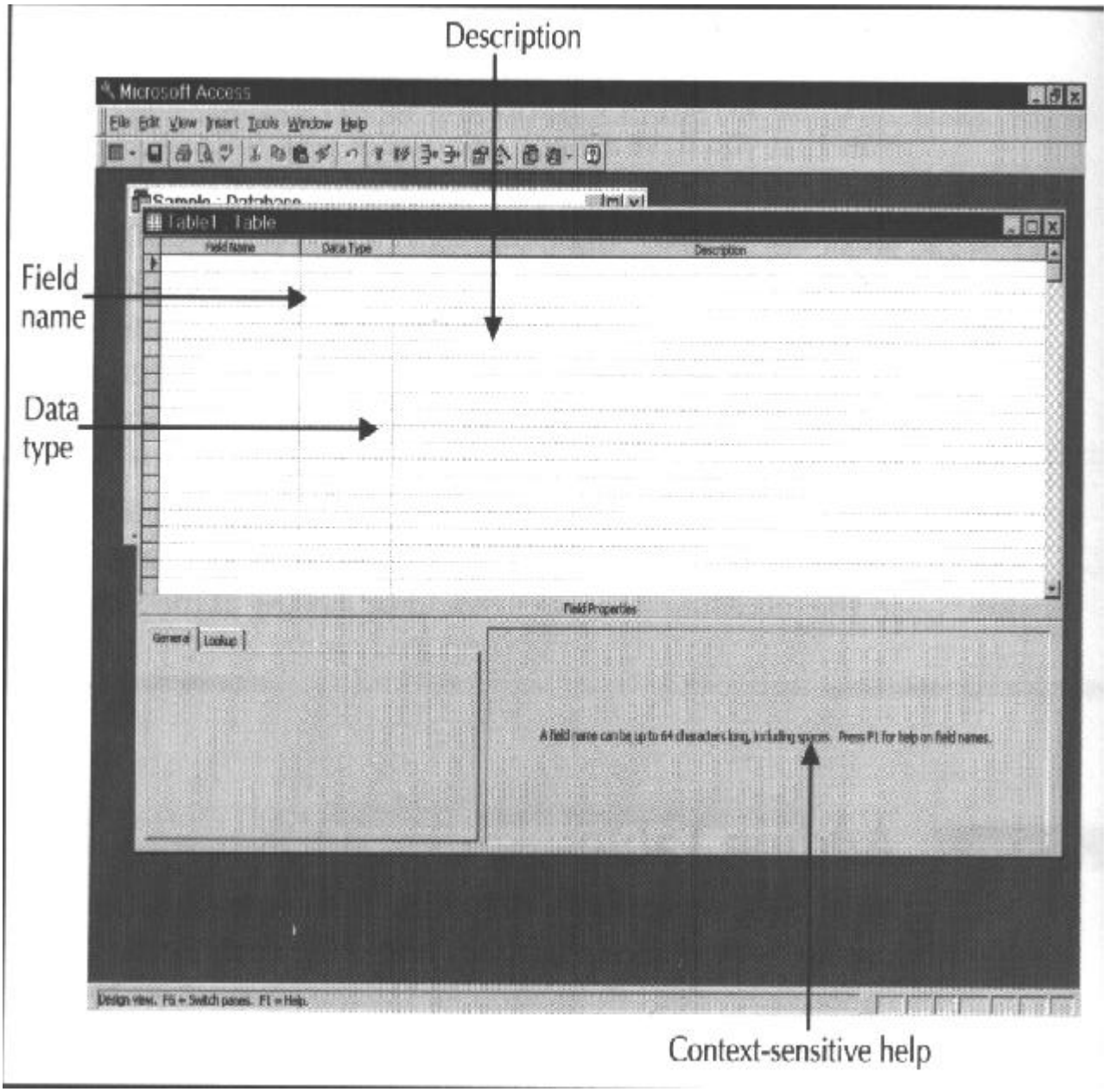


图 5.5　Table（表）设计对话框让你创建示例数据库所需的表

我们花几分钟讨论一下 Table（表）设计对话框。**在这个表中有三列。第一**

列 Field Name（字段名）包含字段的名称。在这里需要填入的只是描述这个字段将保持信息类型的名称。虽然 Access 允许在字段名中使用空格，但要避免这样去做，以便 Visual C++能够更加容易地访问该字段。在字段名中单词间添加空格的最佳方式是，使用像这样的下划线：Field_Name。还要放弃使用任何非字母数字字符，如磅符号(#)或和符号(&)——使用这些字符肯定会导致 C++程序发生故障。

第二列 Data Type（数据类型）定义字段将要保存数据的数据类型。Access 定义了大量的数据类型，各数据类型如下所示。

注释 下列数据类型反映了 Office 97 版的 Access 中所有的数据类型。你会发现，较早版的 Access 提供的可供选择的数据类型要少一些。我们不会在示例数据库中使用任何更新的数据类型，所以较早版的 Access 应该可以良好地运行。

- Text（文本） 包含 255 个或更少字符的短字符串。
- Memo（备忘录） 用于存储注释或其它大段文本的长字符串。
- Number（数字） 任何数值值，包括 Byte（字节），Integer（整数），Long Integer（长整数），Single（单精度实数），Double（双精度实数）和 Replication ID（复制 ID）。

注释 C++程序员会通过术语 GUID（全局唯一标识）了解 Replication ID（复制 ID）。在 C++中，Replication ID 用于识别 Access 对象。

- Date/Time（日期/时间） Access 中几种日期和/或时间格式中的一种。

- Currency（货币） 用来存储货币值的特殊数字格式。
- AutoNumber Access 可以提供一个计数器作为表的主键。主键确保每一个记录都有唯一的可以用来访问它的值，所以创建主键是必要的。
- Yes/No C++程序员会把它作为 Boolean（布尔）值来识别。可以把它格式化为 Access 中的 Yes/No，On/Off 或 True/False。但是，格式化该值的能力并不影响它仍是一个 Boolean（布尔）值这一事实。
- OLEObject（OLE 对象） 一种由主机支持的 OLE 对象。
- Hyperlink（超链接） 这种数据类型最近才被添加到 Access 中。它允许你存储去往网络资源的 UNC（统一命名协定）路径或去往 Internet 资源的 URL（统一资源定位器）。Access 甚至支持在 Web 网页上使用锚地。关于 URL 和锚地的详尽讨论，请参见第 8 章。
- Lookup Wizard（查找向导） 在有些情况下，你想让用户从特定的选择列表中选择，而不是把值输入到表中。Lookup Wizard（查找向导）允许你创建对数据库中另一个表的引用，用来填充这个字段。用户会在屏幕上看到一个列表框或组合框，可以在第二个表的预定义选择列表中选择。

Table（表）设计对话框的第三列 Description（描述），允许你向表中添加注释，以便说明字段的作用。当数据库变得很大时你就会发现，以文档化方式说明数据库每个表中各个项的准确目的是极为必要的。有时候你会发现，起初设计数据库时合理的字段不再需要了。你会发现，如果没有正确的文档，要定位这些消耗额外空间的字段就困难多了。

在 Table（表）设计对话框的左下角，你会看到一个被称为 Field Properties（字段属性）的带选项卡的对话框。在这里输入表的 Access 专用格式化信息。在很多情况下，会看到极小化了的 Caption（标题）属性。Caption（标题）属性定义了 Access 如何在窗体和报告中说明相应的字段。你还会发现，设置标题使设计应用程序变得稍微容易了一些，因为它促使你在设计工程的时候就考虑了用户界面问题。

就在 Table（表）设计对象中 Field Properties（字段属性）的右边，有一个凹下的方框。这个框通常包含了有用的提示信息。例如，如果单击 Field Name（字段名）列，这个框会告诉你，包括空格在内字段名最多可以有 64 个字符长。

现在我们已经熟悉了 Table（表）设计对话框的工作方式，那就来创建两个表把。下图是 Foods（食品）表中包含的字段（简单地输入图中所示的各项，图中还显示了应该使用哪一个数据类型）。

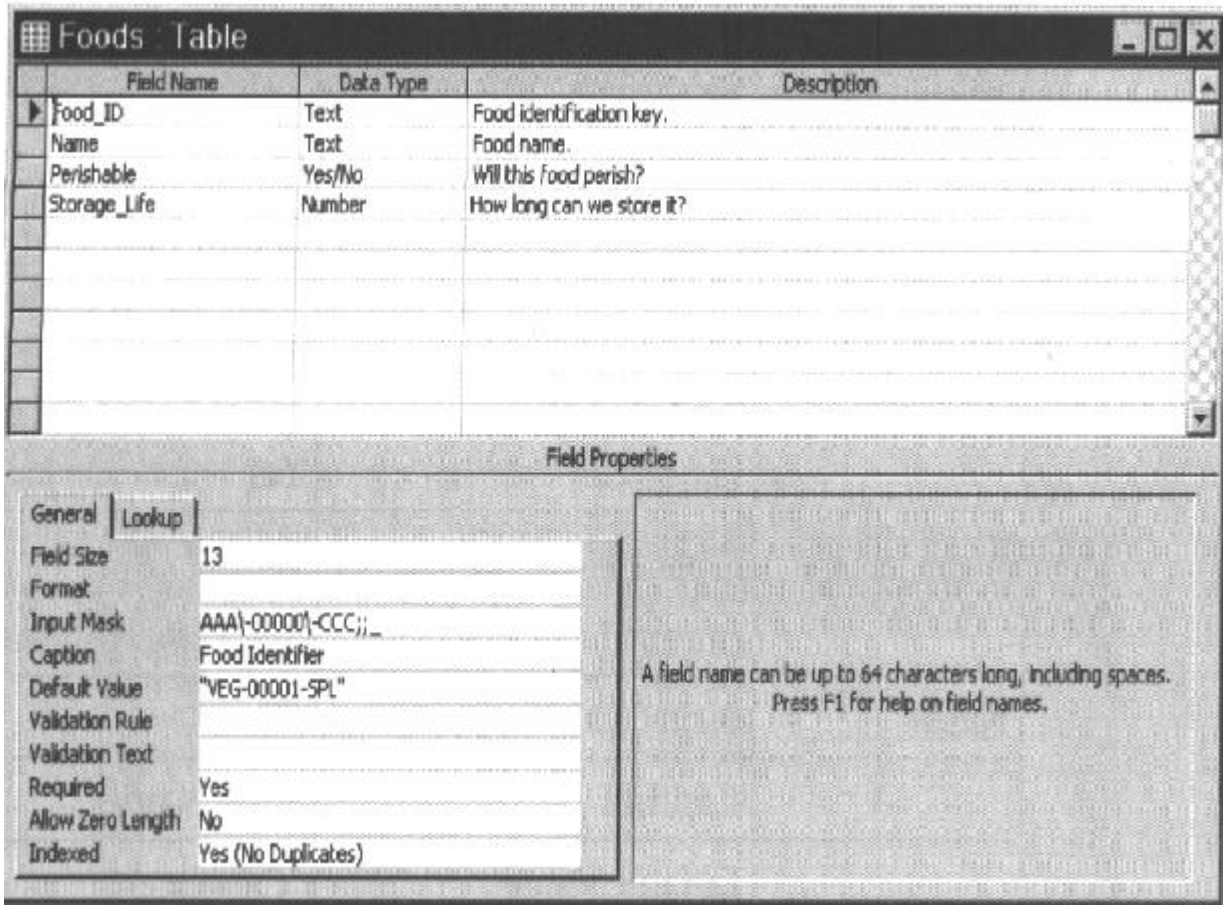


表 5.1 **包含了需要为每一个字段输入的属性值。**

在完成这张表之前，我们还要做上两件事。要做的第一件事是定义主键。我们要用 Food_ID **字段来建立** Foods **（食品）表和** Orders **（订货）表的关系，所以把它作为我们的主键。采用下面的过程定义表的主键。**

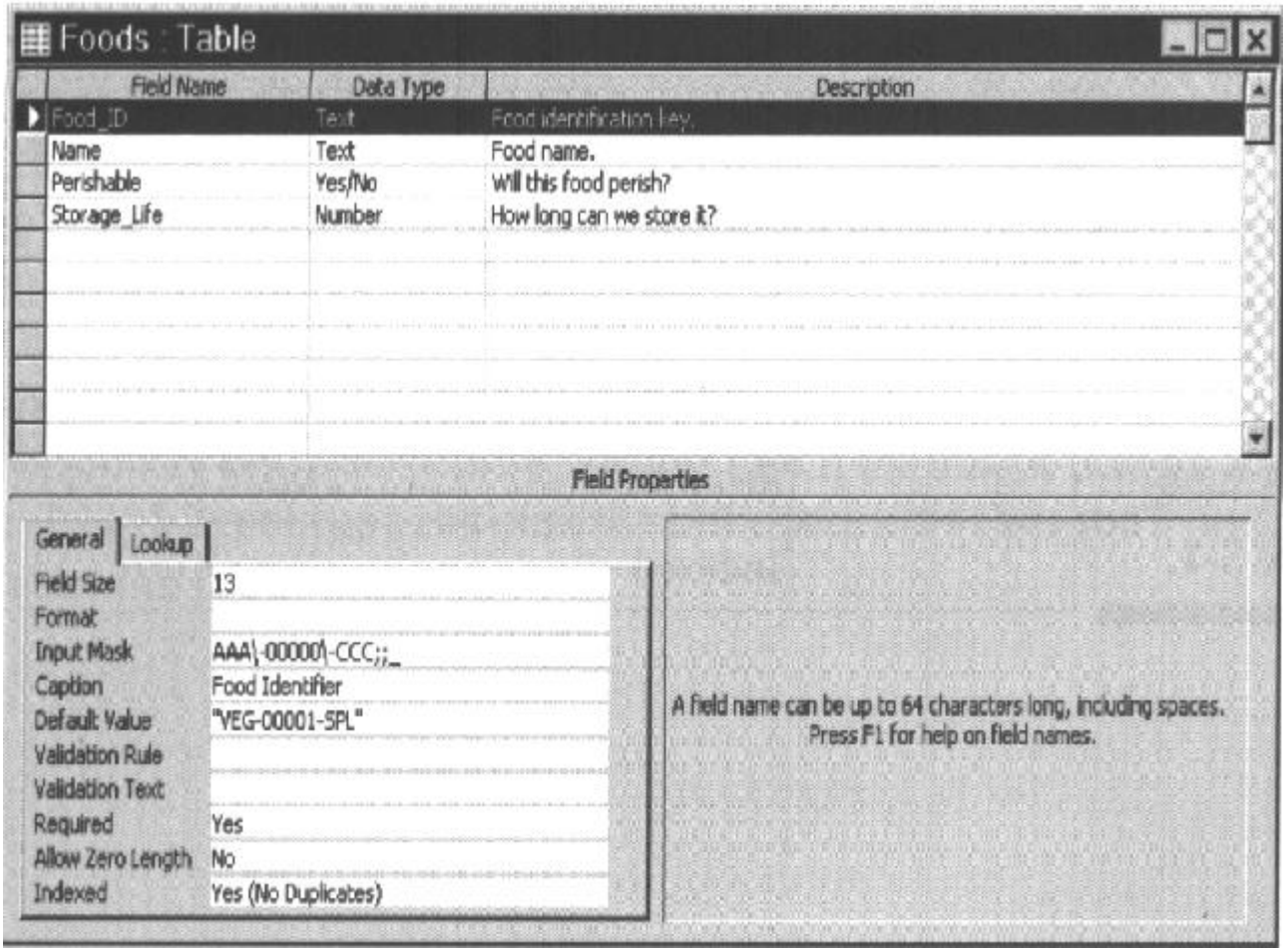
表 5.1　Foods 表的属性值

字段	属性名	值
Food-ID	Field Size	13
	Input Mask	AAA\-00000\-CCC;;_
	Caption	食品标识符
	Default Value	"VEG-00001-SPL"
	Required	Yes（不复制）
	Allow Zero Length	No
	Indexed	Yes（不复制）
Name	Field Size	50
	Caption	食品名称
	Default Value	"卷心菜"
	Required	Yes
	Allow Zero Length	No
	Indexed	Yes（不复制）
Perishable	Caption	易腐坏吗?
	Default Value	True
	Validation Text	必须输入 Yes 或 No
	Required	Yes
	Indexed	Yes（复制完成）
Storage_Life	Caption	保存期限

续表

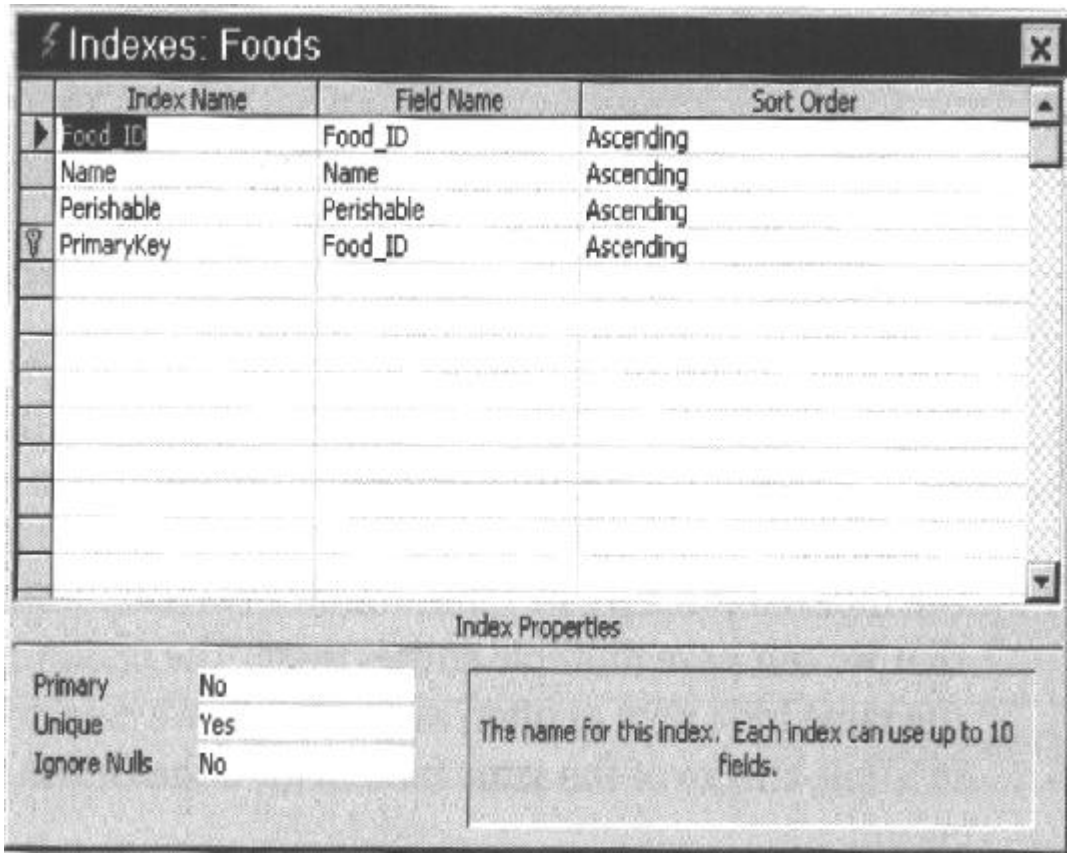
	Default Value	7
	Validation Rule	>0
	Validation Text	值必须大于 0
	Required	No
	Indexed	No

1. 单击 Food_ID 字段左边的方框，整个字段被加亮，如下图所示。



2. 用 Edit|Primary Key（编辑|主键）命令创建主键。Access 会在 Food_ID 字段的左侧添加一个像钥匙一样的小图标。

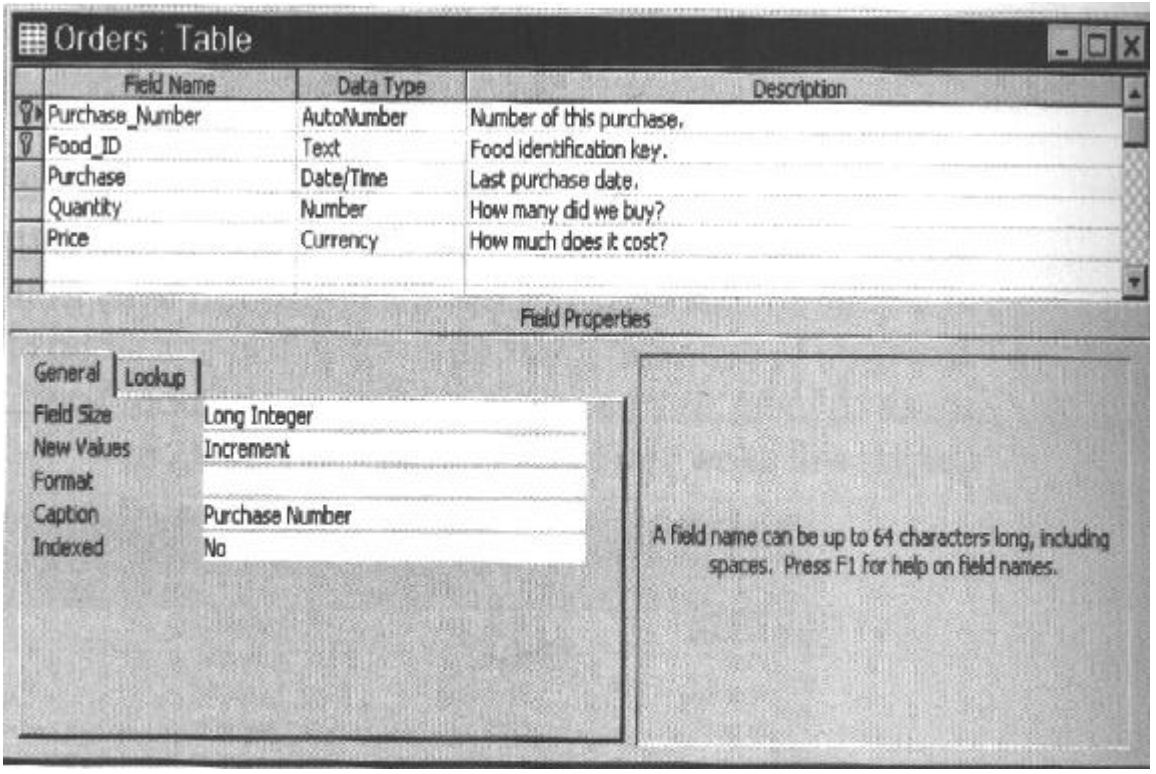
定义了主键后，你会想到要看一看所有索引。用 View|Indexes（视图|索引）命令或单击工具条上的 Indexes（索引）按钮显示 Indexes（索引）对话框，如上图所示。请注意，Access 允许定义单个字段或一组字段的索引；但不能用计算域或等式来创建索引。你会在后面看到，使用查询可以突破这个限制。



单击 Indexes（索引）对话框上的 Close（关闭）框将其关闭。现在保存我们的第一张表。单击工具条上的 Save（保存）按钮。Access 显示一个对话框，要求输入表名。输入 Foods，然后单击 OK。表名会出现在 Sample:Database 对

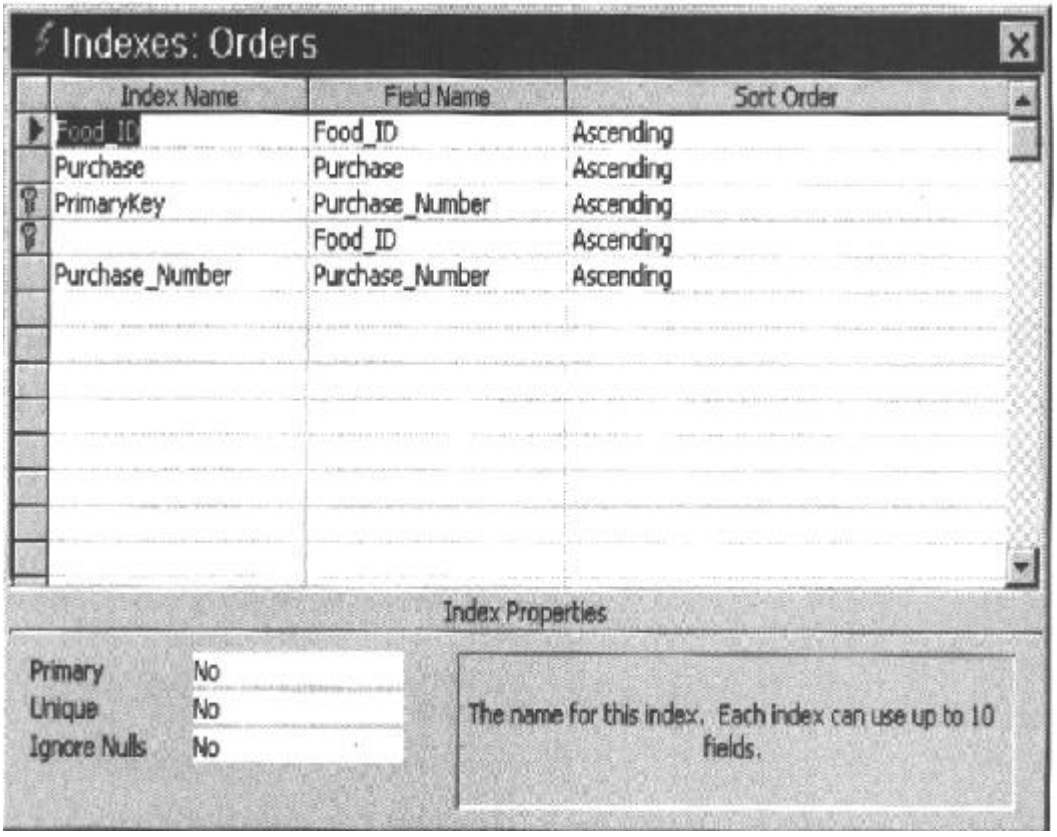
话框中。单击 Close（关闭）框关闭 Table（表）设计对话框。

我们用相同的过程创建第二张表。在此情况下，表名是 Orders。下面是 Orders 表所需的全部字段项（图中还包含了每个字段的数据类型）。请注意，Purchase_Number 和 Food_ID 字段都被用作主键。Purchase_Number 是这张表中取唯一值的字段。表 5.2 包含了完整的需要为这张表定义的属性值。



你应该注意到，表 5.1 中 Food_ID 字段的属性和表 5.2 中相同项的属性存在一个主要的差别。表 5.1 在 Food_ID 字段上建立了索引（参看 Indexed 属性），并且不允许有重复值。原因在于对数据库中的每一个食品来说，我们只想要一个数据项。但是，表 5.2 中的 Indexed 属性值为 Yes，这就是说我们在这里可以有多个重复项。原因很简单：你可能经常需要订购相同的食品，所以我们允许

用户在 Orders **表中使用相同食品的多个重复项。**



我们再检查一下 Indexes（**索引**）**对话框。单击工具条上的** Indexes（**索引**）**按钮。应该看到如下图所示的画面。请注意，我们定义了一个多字段索引** (PrimaryKey)，**方法是，在同一行上输入索引名和第一个字段名，第二行及后续行只包含一个字段名。**

像前面一样，关闭 Indexes **对话框，然后保存新建的表。表名一定要是** Orders。

现在已经有两张表可以使用了，下面开始填充一些数据。图 5.6 **显示了** Foods **表的数据，图** 5.7 **显示了** Orders **表的数据。如果你喜欢的话，可以再添加一些其它数据；只是要保证，对** Foods（**食品**）**表中创建的每一个项，在** Orders（**订**

货）表中都要有一个或多个对应的项。同样地，要保证不把任何在 Foods（食品）表中没有对应项的记录输入 Orders 表中。

表 5.2　Orders 表的属性值

字段	属性名	值
Purchase_Number	Field Length	长整数
	Caption	采购编号
	Indexed	Yes
Food_ID	Field Size	13
	Input Mask	AAA\-00000\-CCC;;_
	Caption	食品标识符
	Default Value	"VEG-00001-SPL"
	Required	Yes
	Allow Zero Length	No
	Indexed	Yes
Purchase	Format	中间日期格式
	Caption	最近一次采购的日期
	Default Value	Now()
	Required	Yes
	Indexed	Yes
Quantity	Field Size	长整数
	Caption	我们买了多少？

续表

	Default Value	1
	Validation Rule	>0
	Validation Text	**必须大于** 0
	Required	Yes
	Indexed	No
Price	Format	**货币格式**
	Caption	**以美元为单位的价格**
	Default Value	0
	Validation Rule	>0
	Validation Text	**必须大于** 0
	Required	Yes
	Indexed	No

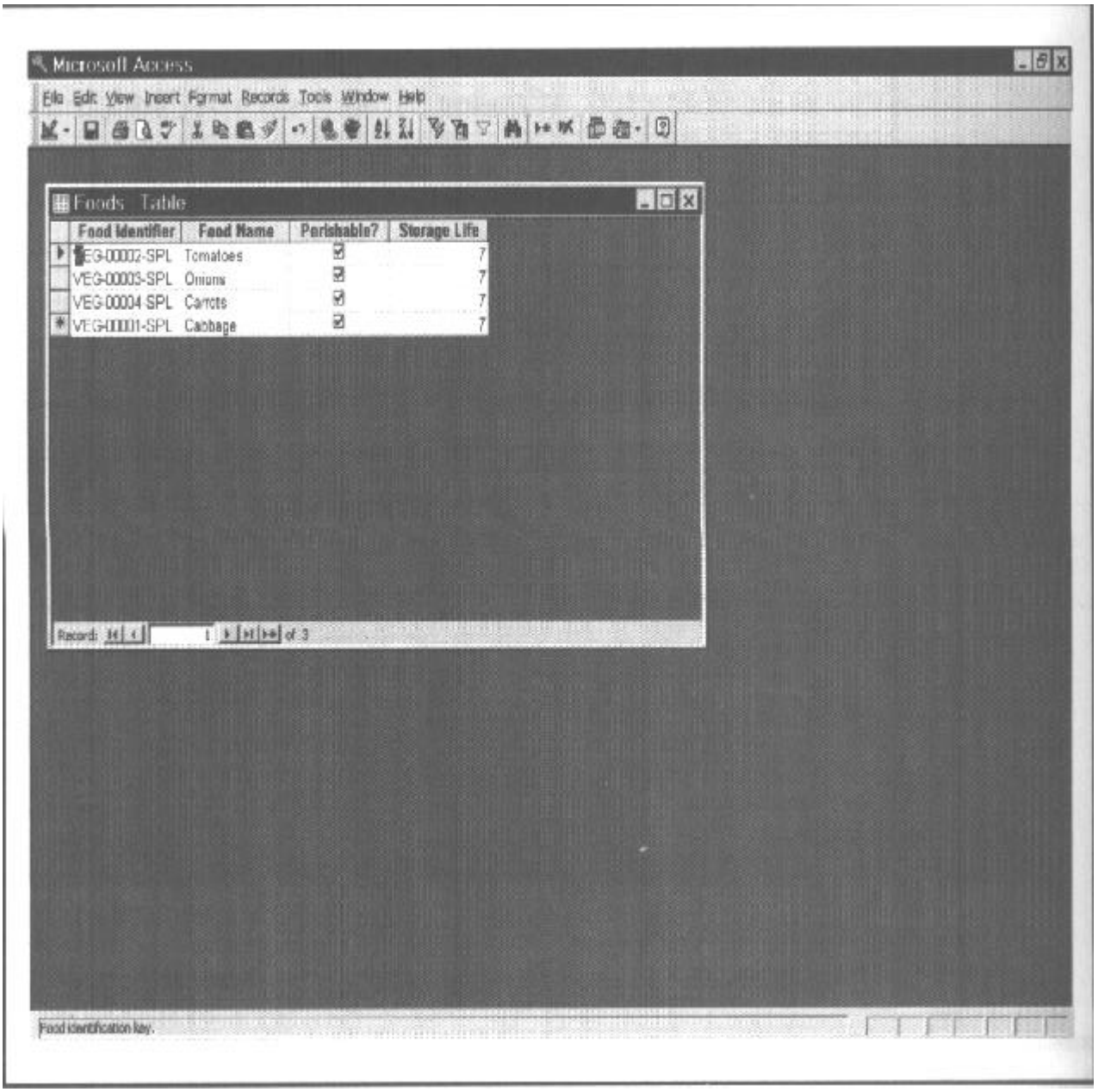


图 5.6　**对** Foods（**食品**）**表中的每一个项，一定要在** Orders（**订货**）**表中也有一个对应项**

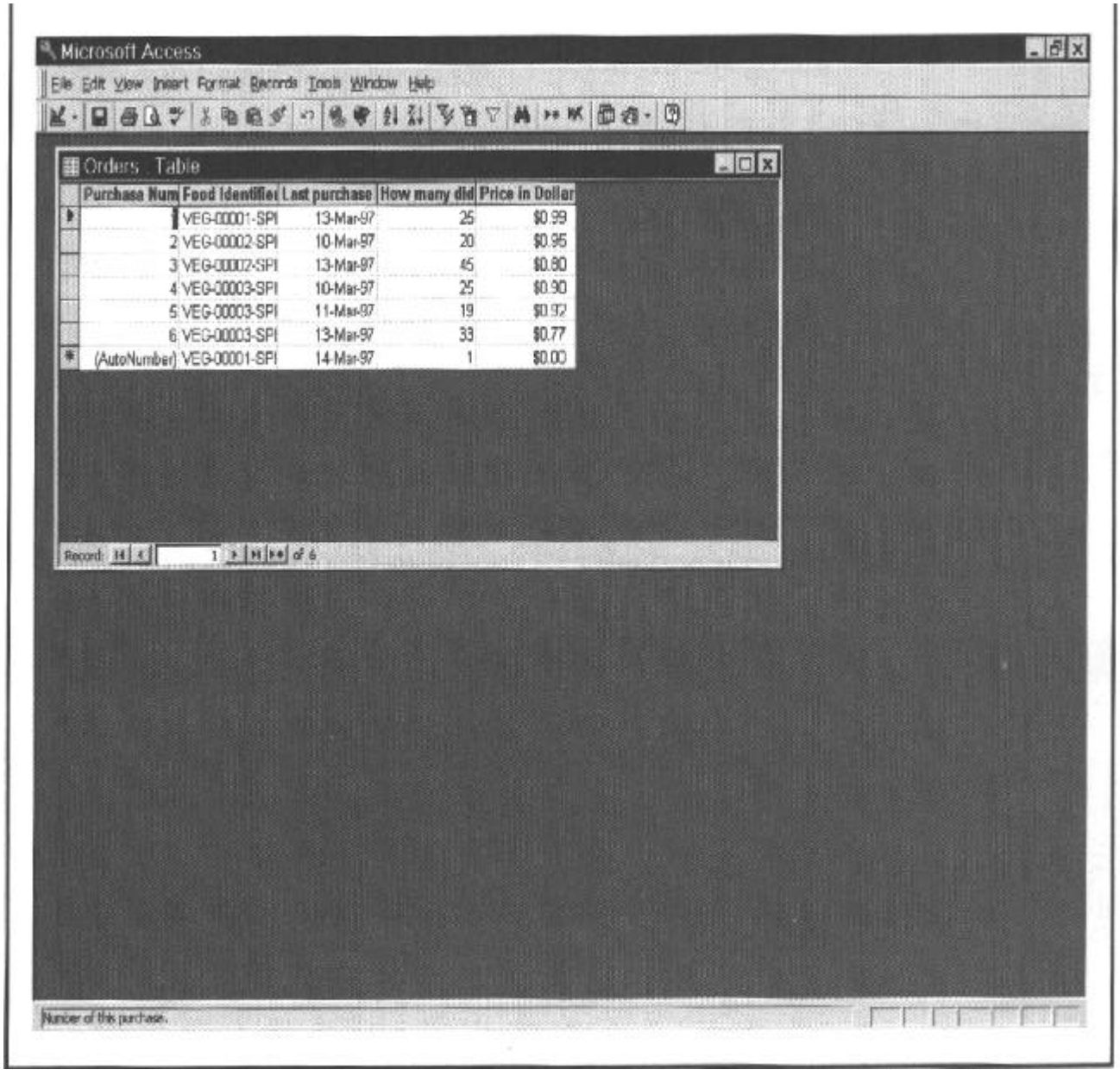
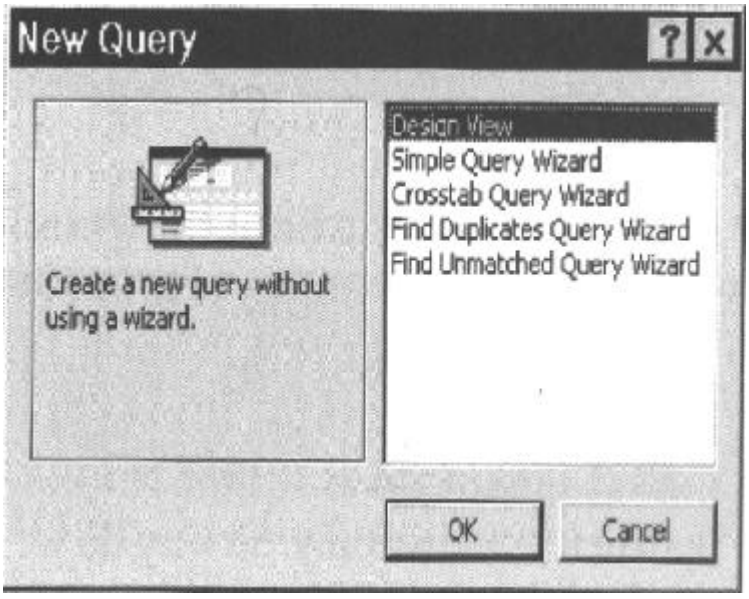


图 5.7　不要在 Orders（订货）表中创建任何在 Foods（食品）表中无匹配 Food_ID 的记录

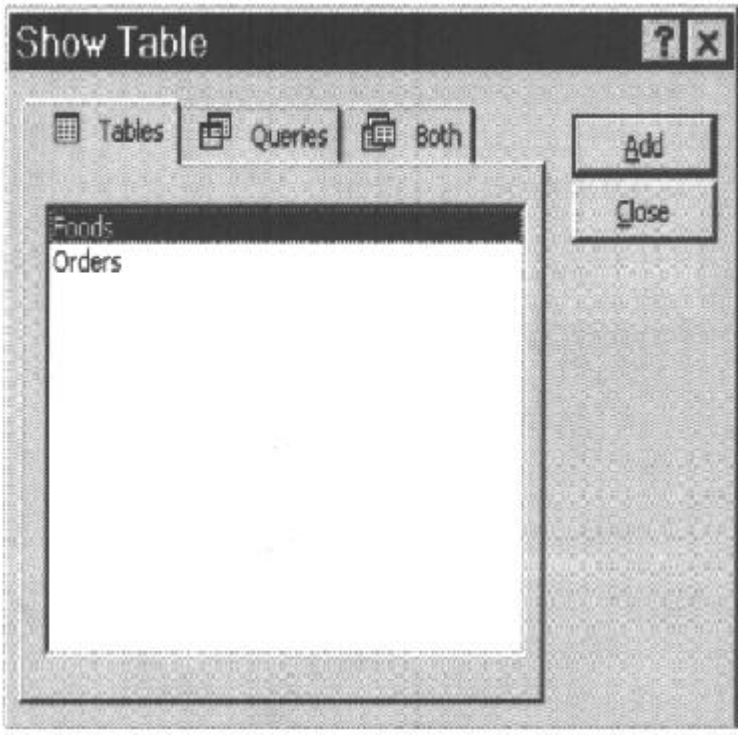
5.4 使用查询排序数据

查询使精确查找所需数据的过程变得令人难以置信的容易。Access 利用可视查询界面使创建查询的过程变得更加容易。下面我们来看一看，为刚刚在上一节创建的两张表创建查询的过程。

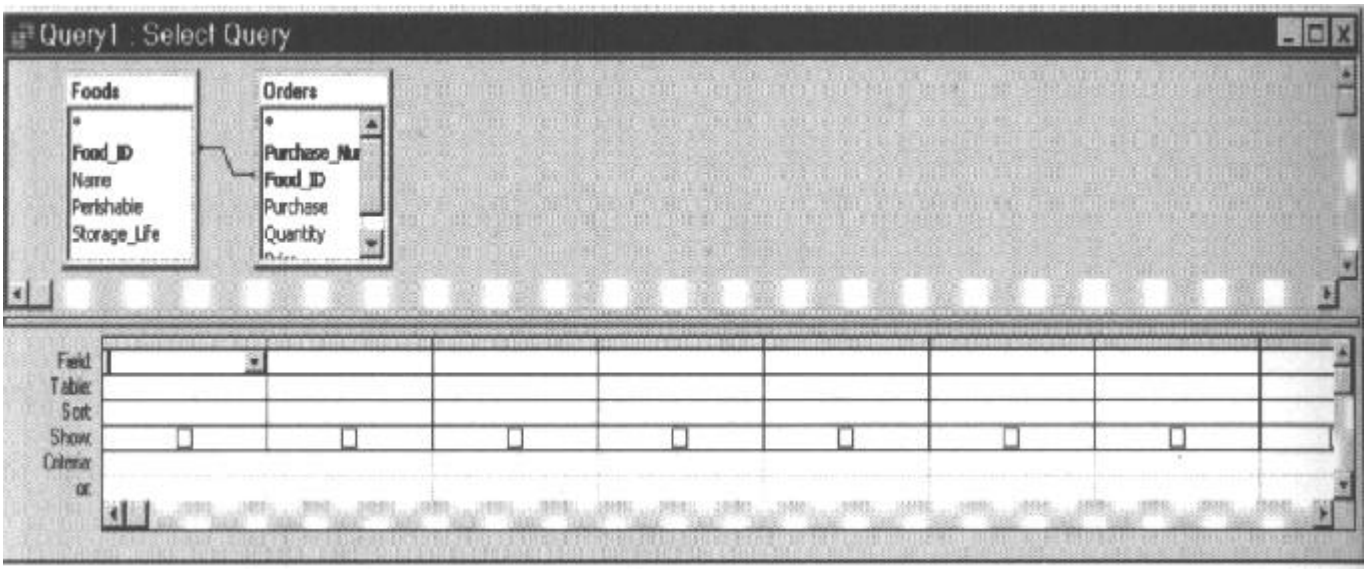
1. 单击 Database（数据库）对话框的 Queries（查询）选项卡。
2. 单击 New（新建），会看到如下图所示的 New Query（新查询）对话框。



3. 选择 Design View（设计视图），然后单击 OK。你会看到如下图所示的 Show Table（显示表）对话框。



4. **加亮这两个表（**Foods **和** Orders**），然后单击** Add**（添加）。单击** Close**（关闭）。系统显示** Show Table**（显示表）对话框，并且你会看到如下图所示的** Select Query**（选择查询）对话框。请注意，**Access **自动创建两个表之间的逻辑关系，主键为粗体字体。现在，我们要决定在查询中显示哪些字段。**



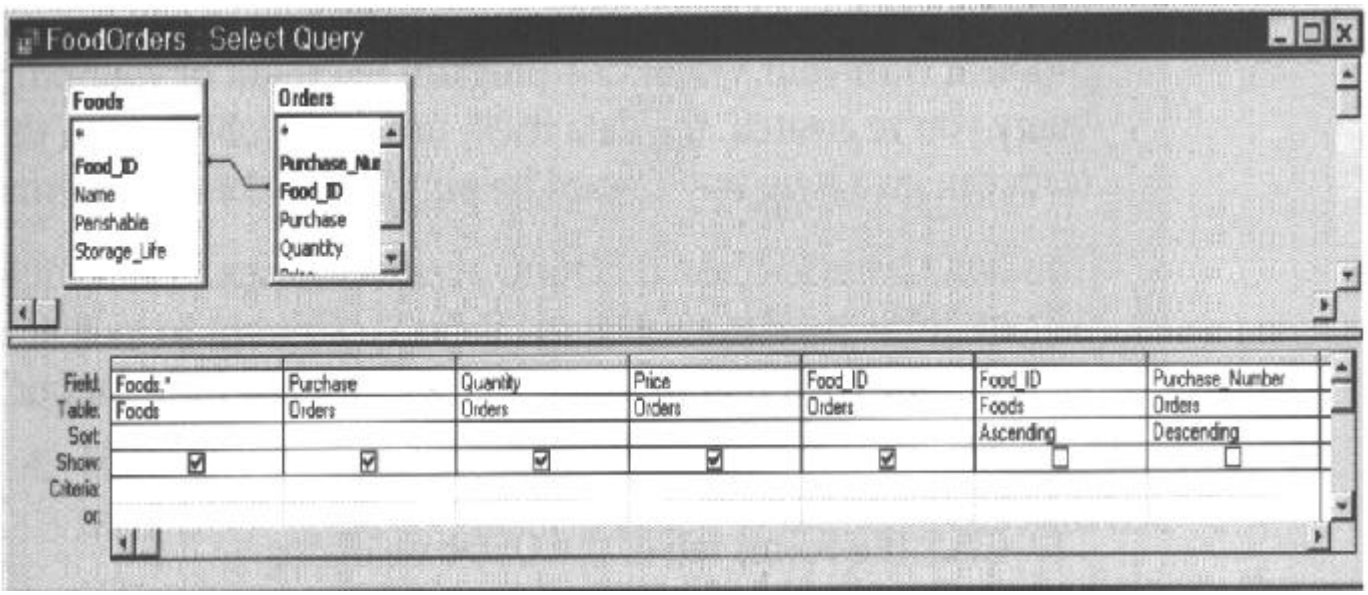
5. **双击** Foods **列表框中的星号。这是告诉** Access**，查询中想使用这个表的所有字段。**

6. **双击** Orders **表中的** Foods_ID**，**Purchase**，**Quantity **和** Price **字段。这是告诉** Access**，查询中我们只想使用这个表中的三个字段。不需要选择** Purchase_Number **字段，因为** Access **会自动更新它。那么，为什么我们需要两个** Food_ID **字段呢？我们要用这个字段来建立两个表之间的关系，所以这样做是很重要的。否则，就不能向** Orders **表中添加记录。现在，我们要决定查询的显示顺序。**

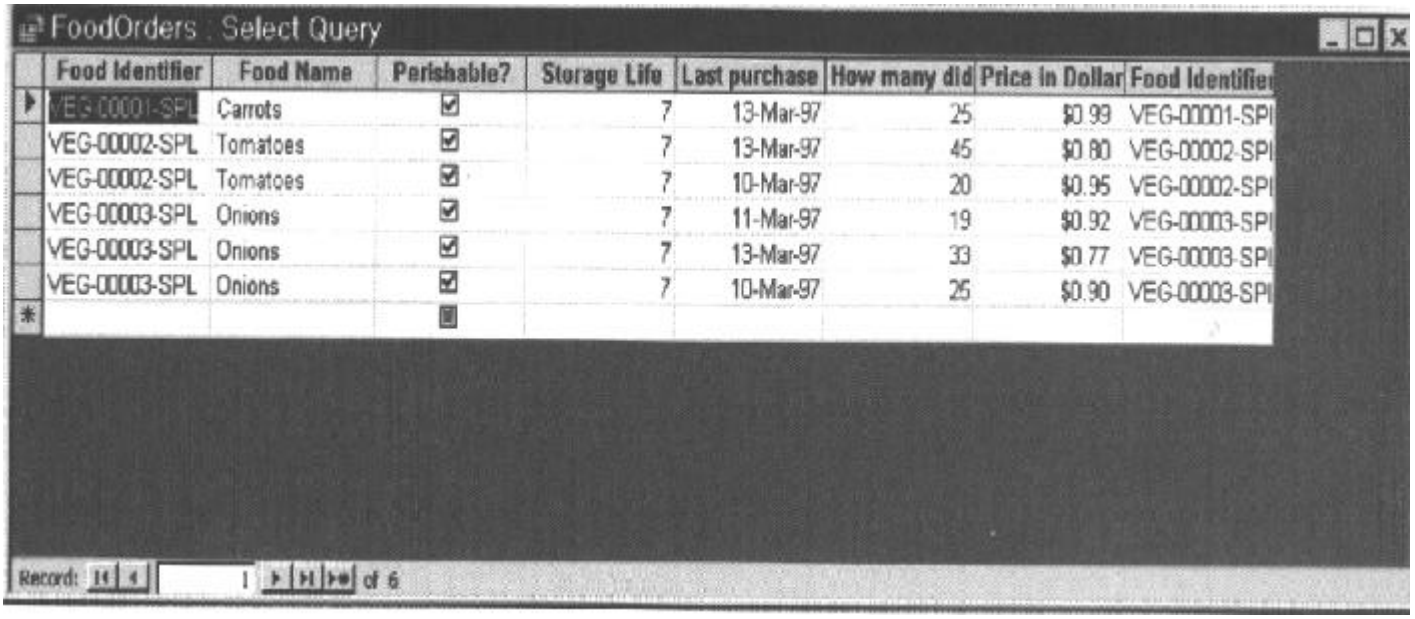
7. **双击** Foods **表中的** Food_ID **字段。在对话框的下半区中，去掉** Show**（显示）属性行中该字段对应的复选标记。我们不想把这个字段显示两次。**

8. **在** Food_ID **字段的** Sort**（排序）属性中选择** Ascending**（升序）。这是告诉** Access **我们想如何排序信息。光用** Food_ID **字段排序还不够，因为我们在** Orders**（订货）表中有这个字段的多个拷贝。**

9. **双击** Orders **表中的** Purchase_Number**，并去掉** Show**（显示）属性的复选标记。在** Sort**（排序）属性中选择** Descending**（降序）。现在，我们有了一种以特别的顺序来显示** Orders **表中所有项的方式。在本例的情况下，首先以最近一次订货的升序方式来显示食品。此时，**Select Query**（选择查询）对话框如下图所示。**



可以测试一下所创建的查询，以确定它们是否按预期方式运行。此时，完成测试的一种最容易的方式是，单击工具条上的 Run**（运行）按钮。你应该看到如下图所示的画面。**



请注意，这次查询中显示了 Orders **表的全部六个数据项，并且只有我们已选择的字段被显示出来。正如我们所要求的那样，所有的食品标识符都以字母数字排序。另外，每项的最新定单显示在前面而不是后面。**

现在保存查询。单击 Save **（保存）按钮，然后在** Save As **（另存为）对话框中输入** FoodOrders**。单击** OK **完成这个过程。单击** Close **（关闭）按钮关闭这个查询。**

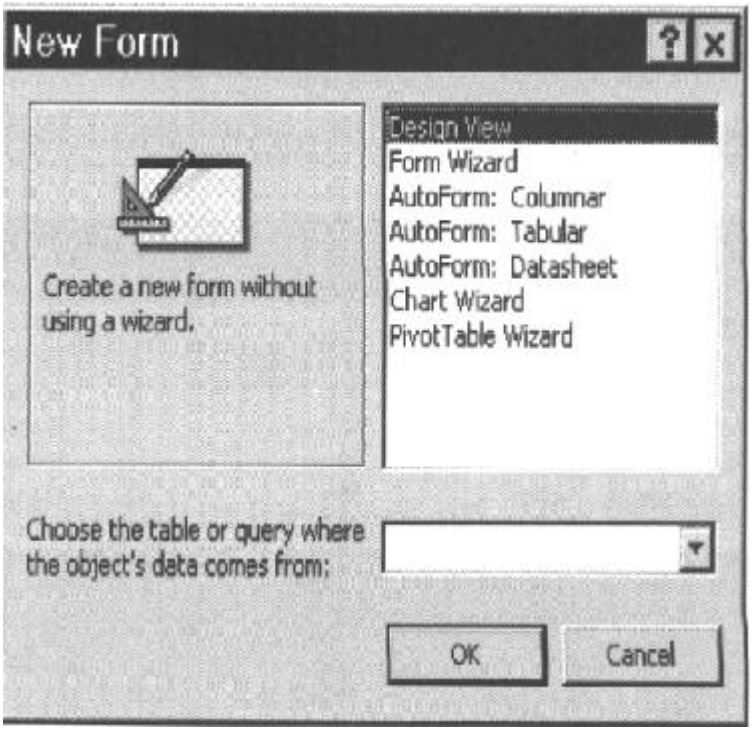
5.5 创建测试窗体

对 Visual C++**程序员来说，在** DBMS **内创建窗体有两个重要的原因。第一，可以在产品类型的环境中再测试一次查询，这个环境和你在** Visual C++**下测试的环境是一样的。第二，测试窗体允许你不必创建特殊的应用程序即能查看数**

据。无论是程序代码、已创建的查询、数据本身或者 DBMS 的故障导致了 Visual C++程序出现错误，你都可以很方便地把错误找出来。

要让窗体完成查询任务，无须把它做得过于复杂。真正需要显示的字段是想在用户应用程序中看到的哪些字段。记住了这一点后，我们用下面的过程为示例数据库创建一个测试窗体。

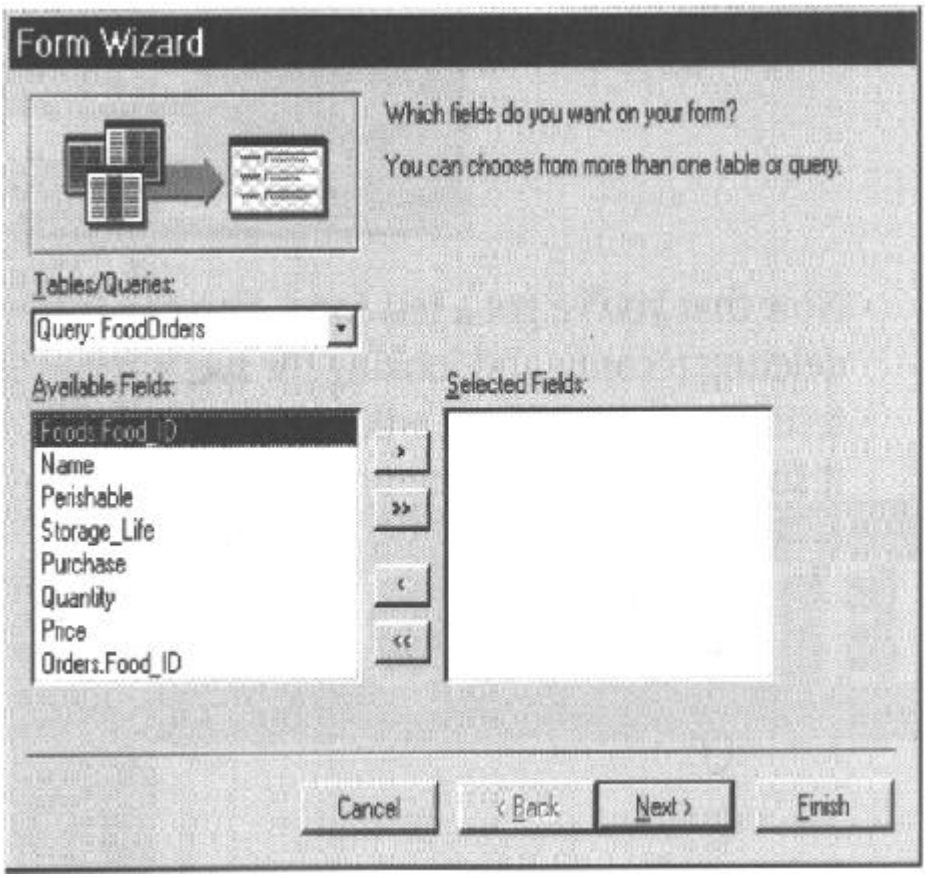
1. 单击 Database（数据库）对话框的 Forms（窗体）选项卡。



2. 单击 New（新建），会看到如下图所示的 New Form（新窗体）对话框。

3. 加亮 Form Wizard（窗体向导），在下拉列表框中选择 FoodOrders 查询，然后单击 OK，会看到如下图所示的 Form Wizard（窗体向导）对话框。仔细看一看字段列表。验证一下 Form Wizard（窗体向导）是否从查询中挑选了你所期

望的全部字段，这一点也很重要。

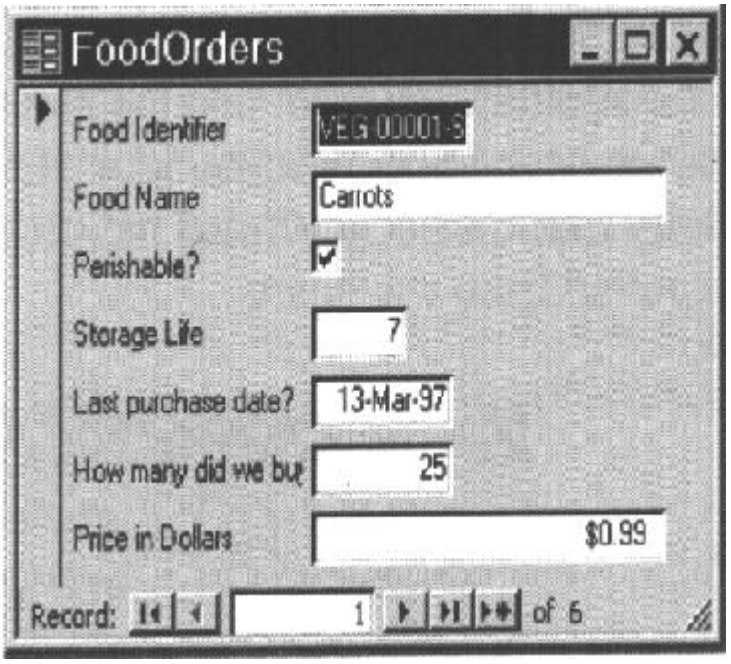


4. **加亮** Orders.Food_ID **字段，然后单击右箭头按钮**(>)**，会看到该字段被添加到** Selected Fields**（已选定字段）列表中。**

5. **单击双箭头按钮**(>>)**，把剩下的字段添加到** Selected Fields**（已选定的字段）列表中。由于我们并不需要** Foods.Food_ID **字段，所以加亮该字段，然后选择左箭头按钮**(<)**，把它从** Selected Fields **（已选定字段）列表中删除。**

6. **由于我们对窗体的美学特性还不感兴趣，所以单击** Finish**（完成）。你会看到数据的单记录视图，如下图所示。你应该再次看一看字段列表，确认所**

需字段都已经出现在窗体上。应该做的第二件事是，验证所选字段的标题都正确无误。最后，验证屏幕上显示的数据是否正确。



现在我们已经有了测试窗体，你也可以试着用窗体添加新记录、删除记录和查找特定的信息，等等。在 Visual C++应用程序使用数据库之前，在实际窗体上进行上述类型的测试，可以帮助你验证是否一切以正常的方式运行。

第 6 章　使用 ODBC 建立数据库应用程序

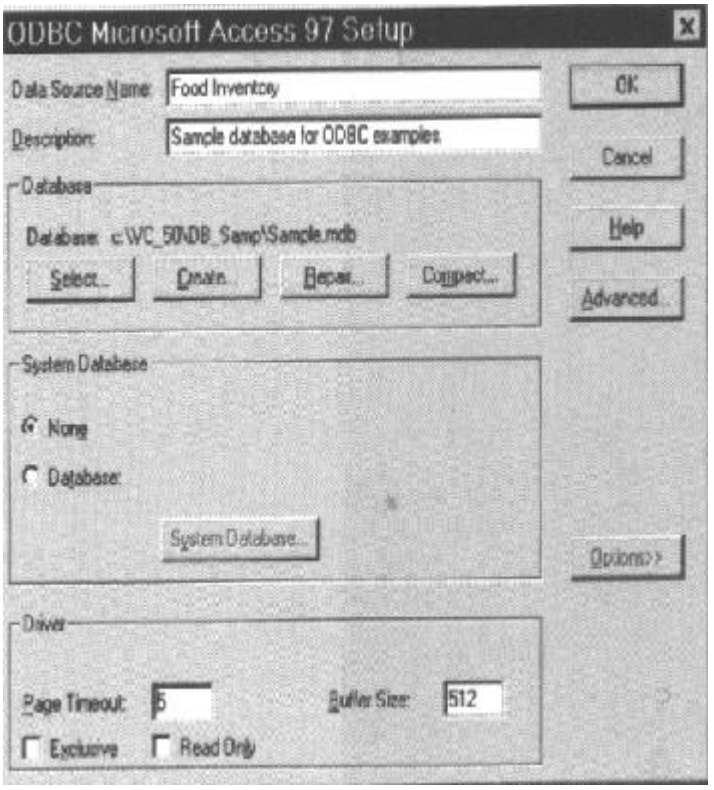
我在前面两章介绍了一些你会在本章中实际用到的概念。我们做的第一件事是，学习一些 ADO，OLE-DB 和 ODBC 理论。此时，假设你决定用 ODBC 来编写应用程序，而不是用 ADO 或者 OLE-DB。如果没有读过第 4 章，现在可能需要补读，这样才有利于下一步的学习。

我们在第 5 章花了一些时间学习如何创建数据库。实际上，本章假定你在那一章中构造了示例数据库，这里就不再讨论示例数据库了，但是我们要在示例代码中使用这个示例数据库。本章的大部分内容只和那两个表和查询有关；如果你确实想跳过示例窗体（第 5 章中说明的窗体）的话，也是完全可以的（但是，示例窗体可以帮助克服表或查询存在的问题）。

Web 链接　谈到使用 ODBC 时，可以花一些时间查阅一下新闻组。以 Access 使用 ODBC 时，一个更为流行的 Microsoft 发起的新闻组是 microsoft.public.access.odbcclientsvr。如果使用了 SQL Server，还可以查阅一下 microsoft.public.sqlserver.odbc 新闻组。当然，除了我们在第 5 章中谈到的更为一般的数据库新闻组外，还有其它许多专门讨论 ODBC 的新闻组。

在开始看一些示例 ODBC 数据库应用程序之前，还有最后一件事要做。回

忆一下第4章,在Visual C++能够实际处理示例数据库之前,我们需要一个ODBC源。可以用第 4 章“使用 ODBC”一节中的过程创建数据源名称(DSN)。下面讲述本章所用的DSN参数。



技巧 在开始编写使用ODBC的Visual C++程序之前,要设置所有的DSN。否则的话,以后你将不得不重新编写某些代码。甚至在启动Visual C++之前就可以完成全部所需的设置,这样可以让MFC AppWizard为你做尽可能多的工作。

请注意,我把Data Source Name(数据源名称)字段设置为"Food Inventory",Description(描述)字段设置为"Sample database for ODBC examples."。还要注

意在本章示例中使用的数据库的位置和名称。为了适应系统的设置，可能需要修改这些设置；用第 4 章阐明的 Select（选择）按钮选择示例数据库。最后，把选项设置成缺省值，不要更改 Set Advanced Options（设置高级选项）对话框上的任何设置。除非设置了数据库安全保护，否则不需要添加缺省用户名或口令——我们在第 5 章中没有设置安全保护。

技巧 一定要以计划使用的 DBMS 测试 ODBC 应用程序，然后使客户知道可能存在的应用程序限制。一些程序员用一个版本的 DBMS 开发应用程序，得到了有趣的结果，一些客户在相同 DBMS 兼容版本的产品环境中也做了一些尝试。例如，一些 Office 97 的用户突然发现，其应用程序的 ODBC 访问次数增加了四倍之多，这使得应用程序事实上不可用（并不是每个人都会碰到这种问题，但它确实给出了一个很好的例子，即在正确的情况下可能发生什么）。即使应用程序是按预想方式工作的，把 Access 的 Office 95 版更新成 Office 97 版也会产生这样预想不到的副作用，即增加了访问次数。这个应用程序的开发者可能是这样避免潜在的问题的，他保证使客户理解其应用程序只是在测试状态下使用 Office 95，或者他用 Office 97 来测试应用程序以寻找存在的副作用。

6.1 创建简单的窗体视图应用程序

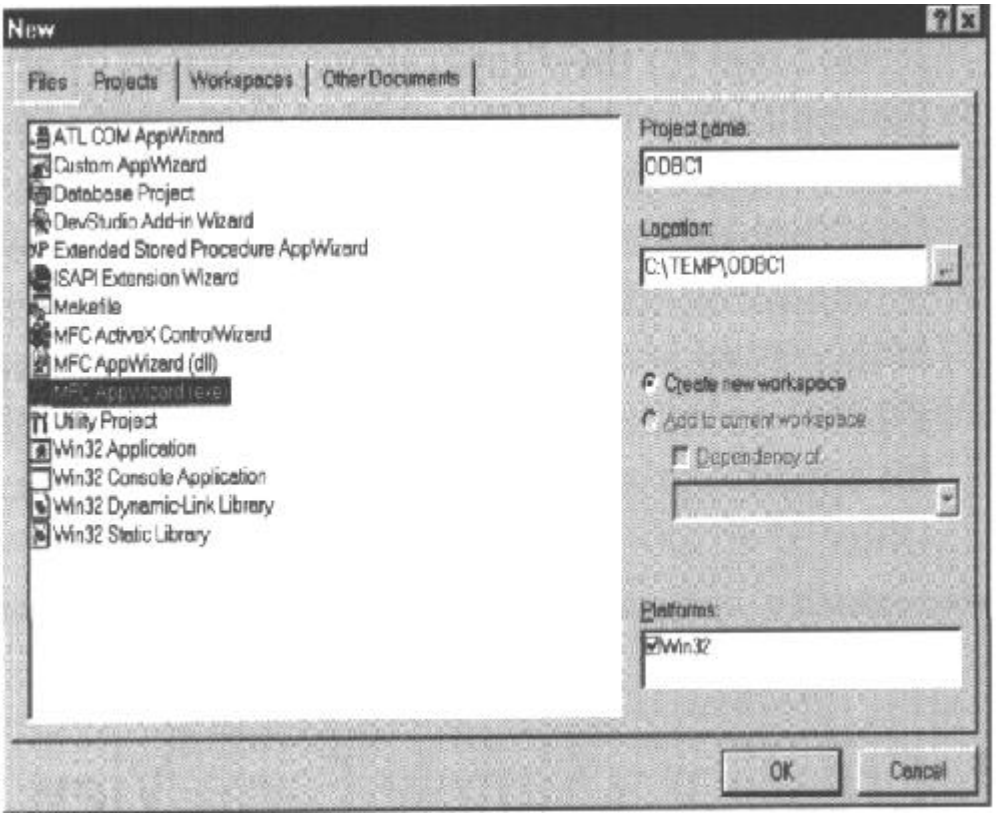
我们从创建最容易的数据库应用程序类型开始，用 Visual C++创建在第 5 章中创建的 FoodOrders 查询的窗体视图。使用查询是启动应用程序并使其运行的最快的方式，因为查询允许你选择想要查看的字段并给数据排序，根本用不着任何代码。下面是创建基本数据库的过程。我们在第 2 章中讲了用 Visual C++ 创建基本单文档应用程序的要点，这里就不再详细研究了。

技巧 有一种情况应该使用表而不是查询，这一点十分重要。如果想创建一个实用程序来修复用户以某种方式损坏的数据库，可能需要进行某种单表访问。否则，即使发现了故障，Access（或任何其它好的 DBMS）所执行的引用完整性规则也可能要阻止你实际修复故障。尽管如此，通常情况下，应该避免更改单独的表，原因在于当应用程序正在运行且引用完整性规则生效时，单表操作的后果难以预料。

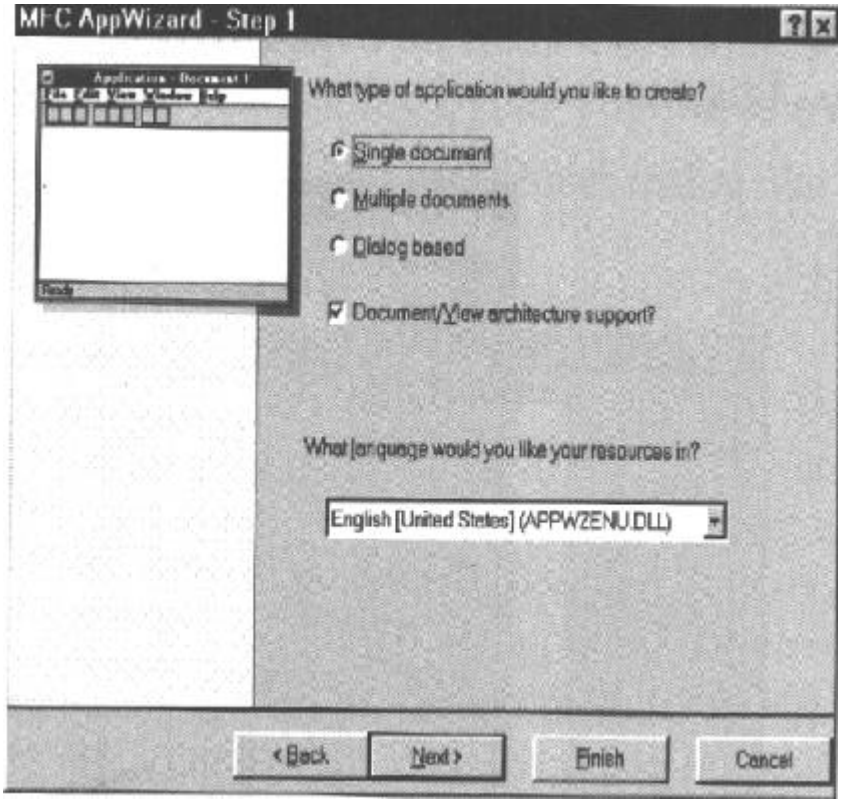
1. 启动 Visual C++。

2. 用 File|New（文件|新建）命令显示 New（新建）对话框。选择 Projects（工程）选项卡。

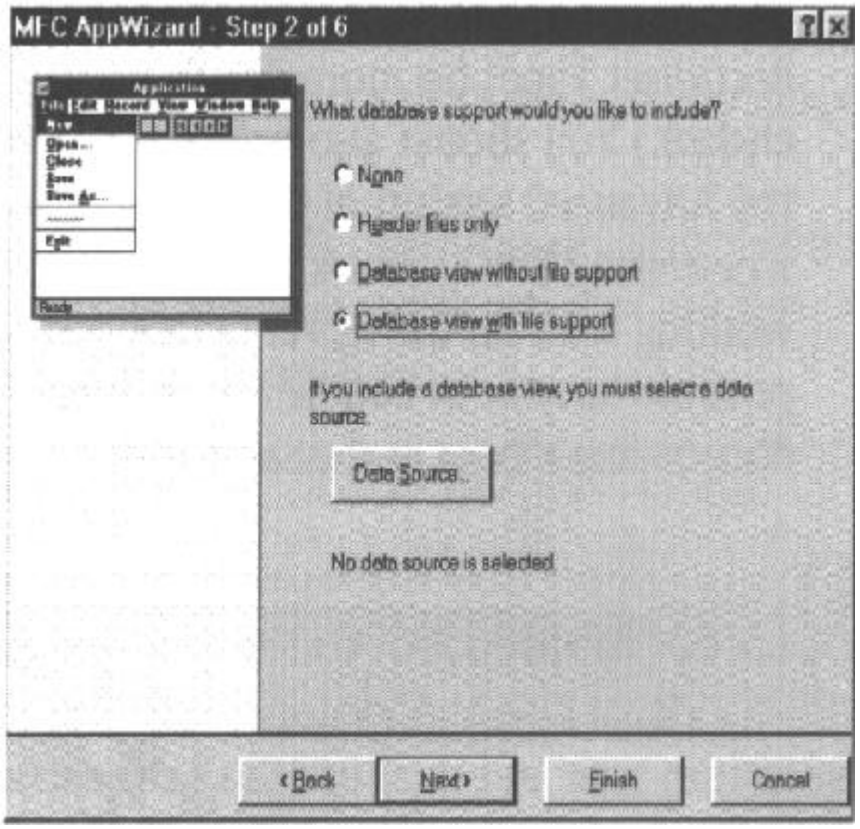
3. 在 Project Name（工程名称）域中输入新的应用程序名（本示例使用 ODBC1），然后加亮 MFC AppWizard(EXE)项，如下图所示。



4. **单击** OK。**会看到如下图所示的** MFC AppWizard-Step1 **对话框。**

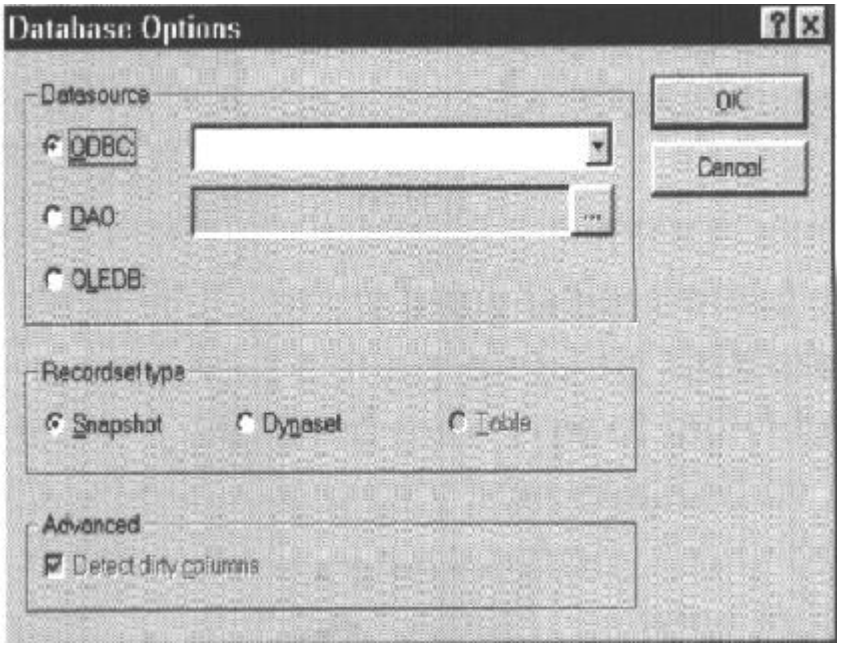


5. **选择** Single Document（**单文档**），**然后单击** Next（**下一个**）。**会看到如下图所示的** MFC AppWizard-Step 2 of 6 **对话框。此时，要选择数据库选项（关于使用** Visual C++ **时数据库选择的详细资料，参见题为“选择数据库选项”的高级技巧框）。**



6. **选择** Database View with File Support **（带文件支持的数据库视图）选项。现在需要选择数据源。请注意，无论选择** Database View with File Support **（有文件支持的数据库视图）选项还是** Database View without File Support **（无文件支持的数据库视图）选项，**Data Source **（数据源）按钮都是可以使用的。**

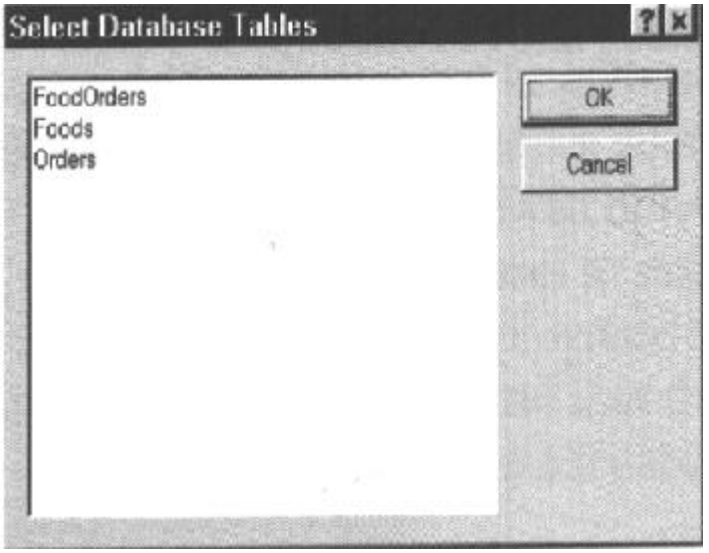
7. **单击** Data Source **（数据源）按钮。会看到如下图所示的** Database Options **（数据库选项）对话框。**



这又是那样一种情形，即在用 MFC AppWizard **设计应用程序的时候，必须确实注意所进行的操作。请注意，这个对话框不但允许你选择数据源，还允许你选择所需的访问类型。我已经在第** 4 **章中谈过了** ODBC、DAO **和** OLE-DB **之间的差别。但是请注意，在选择** ODBC **或** DAO **时还可以选择三类** Recordset（记录集）（关于应该选择的记录集选项的详细资料，参见题为“选择记录集类型”的高级技巧框）。还应该注意对话框的 Advanced（高级）组框中的 Detect Dirty Columns（检测脏列）复选框。这个选项仅在使用 DAO 时是有效的，它允许你在写入更改之前，检测服务器上数据的变化。这就能够处理两个用户试图在同一时间更新同一个记录中数据的问题了。

注释 此时，你可能想知道，如何在 Visual C++应用程序内使用 ADO 呢？你应该选 OLE-DB 作为连接类型，然后在应用程序中使用 ADO 类。这可能有一点让人费解，但当你意识到 OLE-DB 是 ADO 的基础访问技术时，也就明白其中的奥妙了。

8. **选择** ODBC **选项，因为我们会在本章中使用** ODBC **访问。应该在下拉列表框中选择** Food Inventory **（食品库存）列表项——这是我们在本章开始时创建的** DSN**。要想马上看到对数据库所做的更改，还要选择** Dynaset **（动态集）选项。**



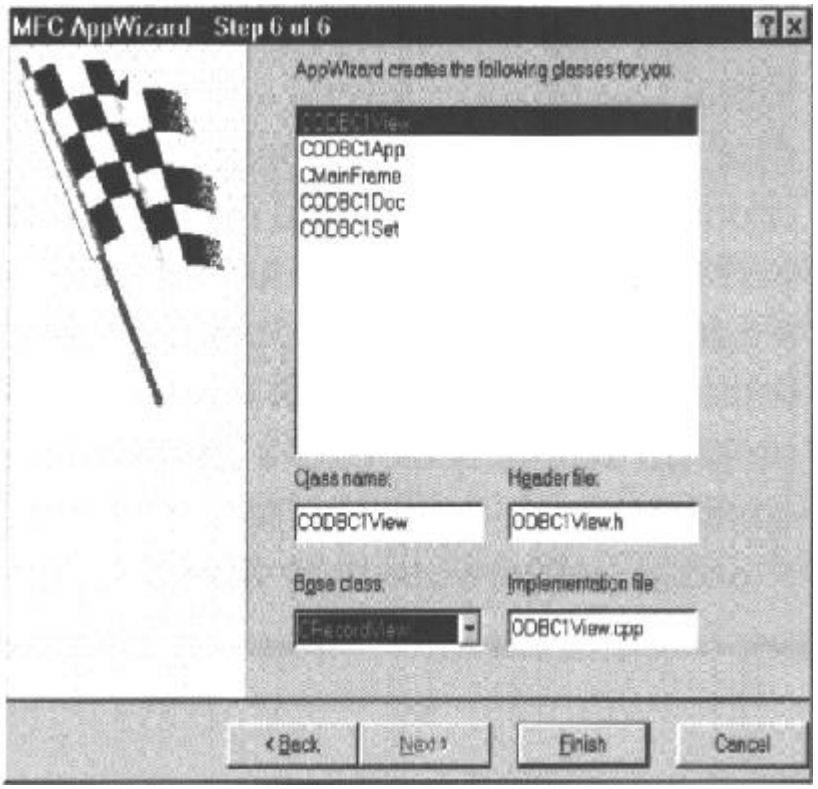
9. **单击** OK**。会看到如下图所示的** Select Database Tables **（选择数据库表）对话框。**

10. **加亮** FoodOrders **（食品订货）选项。这是我们在第** 5 **章中创建的查询。请注意，你也可能选择两个表中的一个，但通常会因为我们在前面谈到的理由**

而选择查询。一定要在这里选择正确的条目，因为今后不会再有第二次选择机会了。和简单地从头编写应用程序相比，因选择错误而产生的工作量更大，如果不仔细的话，会导致代价昂贵的时间延误。

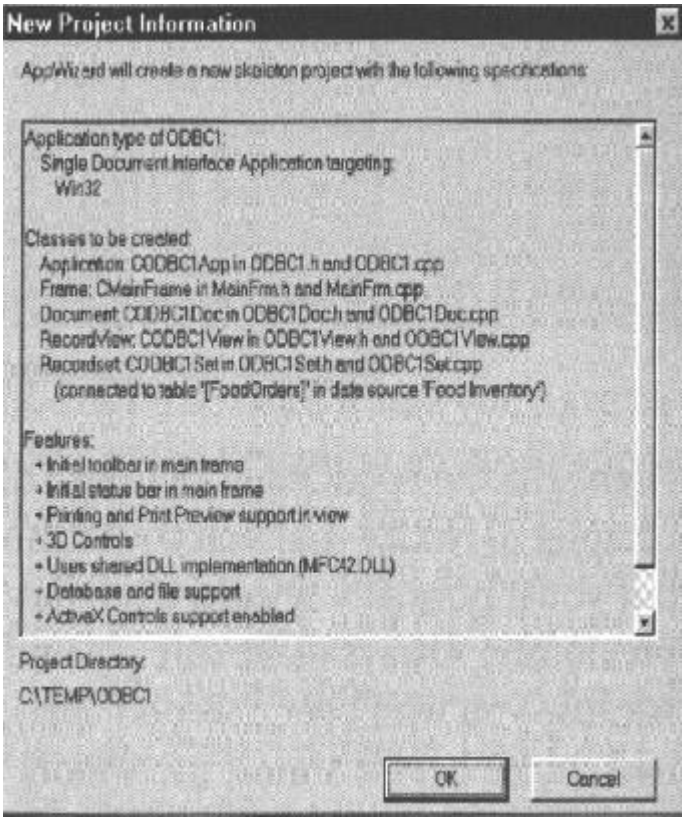
11. 单击 OK 返回 MFC AppWizard-Step2 of 6。此时，已经用我们在第 5 章中创建的示例数据库配置完应用程序，可以使用了。

12. 单击 Next（下一个）四次。应该看到如下图所示的 MFC AppWizard-Step 6 of 6。



13. 单击 CODBC1View。请注意，CRecordView 是应用程序的基类。在后续章节中会发现，无须接受这个缺省值，就像我们在第 2 章单文档应用程序中没有接受缺省值一样。

14. **单击** Finish**(完成)。**Visual C++**显示如下图所示的** New Project Information **(新建工程信息)对话框。**



15. **单击** OK。Visual C++**生成应用程序。**

高级技巧

选择数据库选项

Visual C++提供了几个不同的选项，允许或禁止应用程序中的数据库支持。所选择的数据库支持层次依赖于正创建的应用程序的类型，虽然这并不是需要考虑的唯一标准。例如，你可能需要数据库是应用程序的一部分（像游戏程序），而不需要用户直接和它交互作用。同样地，就所要求的支持层次而言，实用程序肯定和你为安全数据任务所创建的应用程序不同。但是，用户可能需要访问同一数据库的几个不同版本，所以要允许用户从应用程序内把它打开。下面的一览表可以帮助你理解可用的选项。

- None（无） 选择这个选项意味着不想要任何数据库支持。今后再添加它是很烦琐的，所以如果有应用程序需要某种数据库支持的可能性，就请仔细考虑一下这个选项。甚至，非数据库应用程序都要经常利用数据库来存储配置设置或者其它信息。众所周知，游戏程序内含扩展数据库，你还会发现，几乎没有帮助桌面应用程序可以不使用数据库。

- Header Files Only（只要头文件） 如果认为需要数据库支持但确实对其细节不清楚的话，这是应该选择的最好选项。Visual C++向应用程序中添加所要求的头文件，但会要求你自己创建数据库类。实际上，这项工作比想像得要少，因为 Class Wizard 会为你完成创建类的最困难部分的工作。你要做的，只是添加实际检验数据库内容所需的代码。

- Database View without File Support（无文件支持的数据库视图） 有时候，你不想用户在可用文件的外部对文件做任何处理。这个选项要 Visual C++创建访问 ODBC 数据源所需的全部类，你已经提供了类，但不准备制作打开其它文件所需的菜单条目。自定义数据库应用程序的建立者们可能用这个选项用得最多。例如，如果这个应用程序被设计成用来编辑公司的工资表，就不想要用户私自打开文件，破坏安全措施。消除文件支持可能不会使数据库完全安全，但它确实排除了又一个潜在的安全漏洞。

- Database View with File Support（带文件支持的数据库视图） 实用程序总是使用这个选项，因为它为用户提供了最大限度的自由。不但你的应用程序会访问已被设计到应用程序中的数据源，而且你还可以打开外部文件。当然，用户实际使用这个特征的能力要依赖于已建立程序本身的能力。

高级技巧

选择记录集类型

Visual C++提供了几种记录集，可以用来定制应用程序的工作方式。查看这些不同选项的最快方式要兼顾速度和特征。你会发现，在很多情况下，如果想添加特征，就必须付出程序执行速度降低的代价。下面告诉你一些可以自由支配的记录集选项。更重要的是，要告诉你从这个选项可以获得更快的速度还是更多的特征。

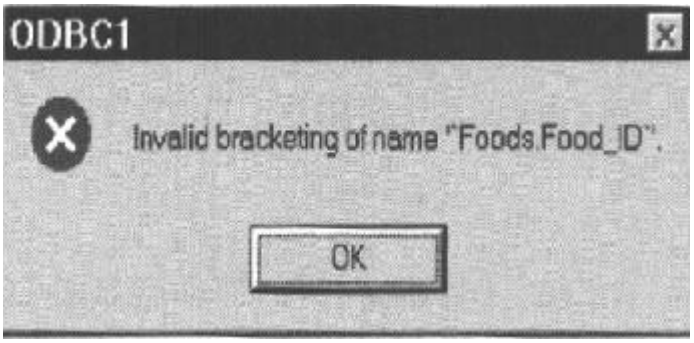
- Snapshot（快照） 这个选项要 Visual C++在一次快照中下载整个查询。换言之，及时快速地给数据库内容拍照，并把它作为未来工作的基础。这种方法有三个缺点。第一，你看不到别人在网络上做的更新，这可能意味着你的决定是建立在老信息的基础上。第二，一次就下载所有这些记录，这意味着在下载期间给网络增加了沉重的负担。第三，记录下载时用户会结束等待，这意味着网络的呼叫性能变得更低。然而这种方法也有两个优点。第一，记录一旦被下载，该工作站所需的网络活动几乎就没有了——这为其它请求释放了带宽。总之，你会看到网络的吞吐量增大了。第二，因为所有被申请的记录都在用户的机器上，所以用户实际上会得到应用程序更佳的总体性能。你可能想把快照的方法限制在较小的数据库上使用，原因在于快照适用于用户请求信息而不适用于数据编辑会话。

- Dynaset（动态集） 使用这个选项时，Visual C++创建指向所请求的每个记录的实际指针。另外，只有填充屏幕时实际需要的记录是从服务器上下载来的。这种方法的好处很明显。几乎马上就能在屏幕上看到记录。而且还会看到其它用户对数据库所做的更改。最后，其它用户也会看到你做的更改，因为动态集在你更改记录时被上载到服务器上。很明显，这种方法要求对服务器的实时访问，它减小了网络总吞吐量并降低了应用程序的性能。这个选项适合于创建用户要花费很多时间来编辑数据的应用程序。同时，它也是大型数据库的最佳选择，原因在于只需下载用户实际需要的信息。

- Table（表） 我们已经考察了记录级的其它两个选项。表方法（仅当使用 DAO 时可用）把所做查询的内容放进一个临时表。这样做不但减小了从服务器下载的信息量，还意味着程序员有了更大的灵活性，因为可以直接操纵临时表字段和记录。你必须为这个选项付出多大代价呢？如果不仔细的话，从服务器下载的过程实际上可能比记录子集花的时间还要长。另外，还有更新方面的问题——你看不到别人所做的更改，他们也看不到你所做的更改。当你正在使用 DAO 且用户会执行同等数量的数据查找和数据编辑时，这可能是最佳的选择。

修复 MFC AppWizard 产生的错误

现在已经有了应用程序的框架，可以编译它了（尽管你没有看到任何数据，但示例应用程序确实发挥作用了）。运行应用程序，你可能会对出现下图所示的错误对话框窗体感到惊奇（你的对话框看上去可能和我的略有不同，这要看 MFC 是连接应用程序的方式以及正在使用的 Visual C++的版本）。



如果没有看到这个消息，意味着 Microsoft 修复了 MFC AppWizard 中的这个问题，你可以跳过本节的剩余部分。错误消息实际上并不是来自应用程序；而是来自 Access，因为 MFC AppWizard 不能正确地处理复杂的字段引用。我们来看一下这个问题并找到最快的解决办法。

双击 CODBC1Set 类的 DoFieldExchange()函数。程序列表 6.1 显示了你将看到的示例。RFX_Text()函数正在把数据从数据库传输到我们的 Visual C++应用程序。请注意，第一个调用操作 Food_ID 字段。你需要把这个字段引用从 "[Foods.Food_ID]"（即有一个名为 Foods.Food_ID 的字段）修改为 "Foods.[Food_ID]"（即在 Foods 表中有一个名为 Food_ID 的字段）。要在最后一个 RFX_Text 调用中做相似的更改。把"[Orders.Food_ID]"引用修改为

"Orders.[Food_ID]"。

程序列表 6.1

```
void CODBC1Set::DoFieldExchange(CFieldExchange* pFX)
 {
 //{{AFX_FIELD_MAP(CODBC1Set)
 pFX->SetFieldType(CFieldExchange::outputColumn);
 RFX_Text(pFX, _T("[Foods.Food_ID]"), m_Food_ID);
 RFX_Text(pFX, _T("[Name]"), m_Name);
 RFX_Bool(pFX, _T("[Perishable]"), m_Perishable);
 RFX_Int(pFX, _T("[ Storage_Life]"), m_Storage_Life);
 RFX_Date(pFX, _T("[Purchase]"), m_Purchase);
 RFX_Long(pFX, _T("[Quantity]"), m_Quantity);
 RFX_Text(pFX, _T("[Price]"), m_Price);
 RFX_Text(pFX, _T("[Orders.Food_ID]"), m_Food_ID2);
 //}}AFX_FIELD_MAP
}
```

重编译应用程序。这一次运行时，就看不到任何错误了。当然，也看不到任何数据。我们会在下一节讨论这个问题。

添加一些数据显示代码

在我们准备向这个示例中添加代码之前，要执行几个步骤。必须做的第一件事是，创建用于显示数据的窗体。图 6.1 显示了我们这次要使用的窗体（可以根据个人爱好来修改它；只要保证所有字段都齐全就行了）。

对窗体上的各种控件，我使用了下面的 ID（从上到下）——它们与表所使用的字段值相对应：IDC_FOOD_ID，IDC_NAME，IDC_PERISHABLE，IDC_PRICE，IDC_PURCHASE，IDC_QUANTITY 和 IDC_STORAGE_LIFE。需要把 IDC_PRICE 控件的 Align Text（对齐文本）属性设置为 Right（右对齐），并选中 Multiline（多行）属性。对另外两个数字域，你也可以把它们的 Align Text（对齐文本）属性设置为 Right（右对齐），并选中 Multiline（多行）和 Number（数字）属性。

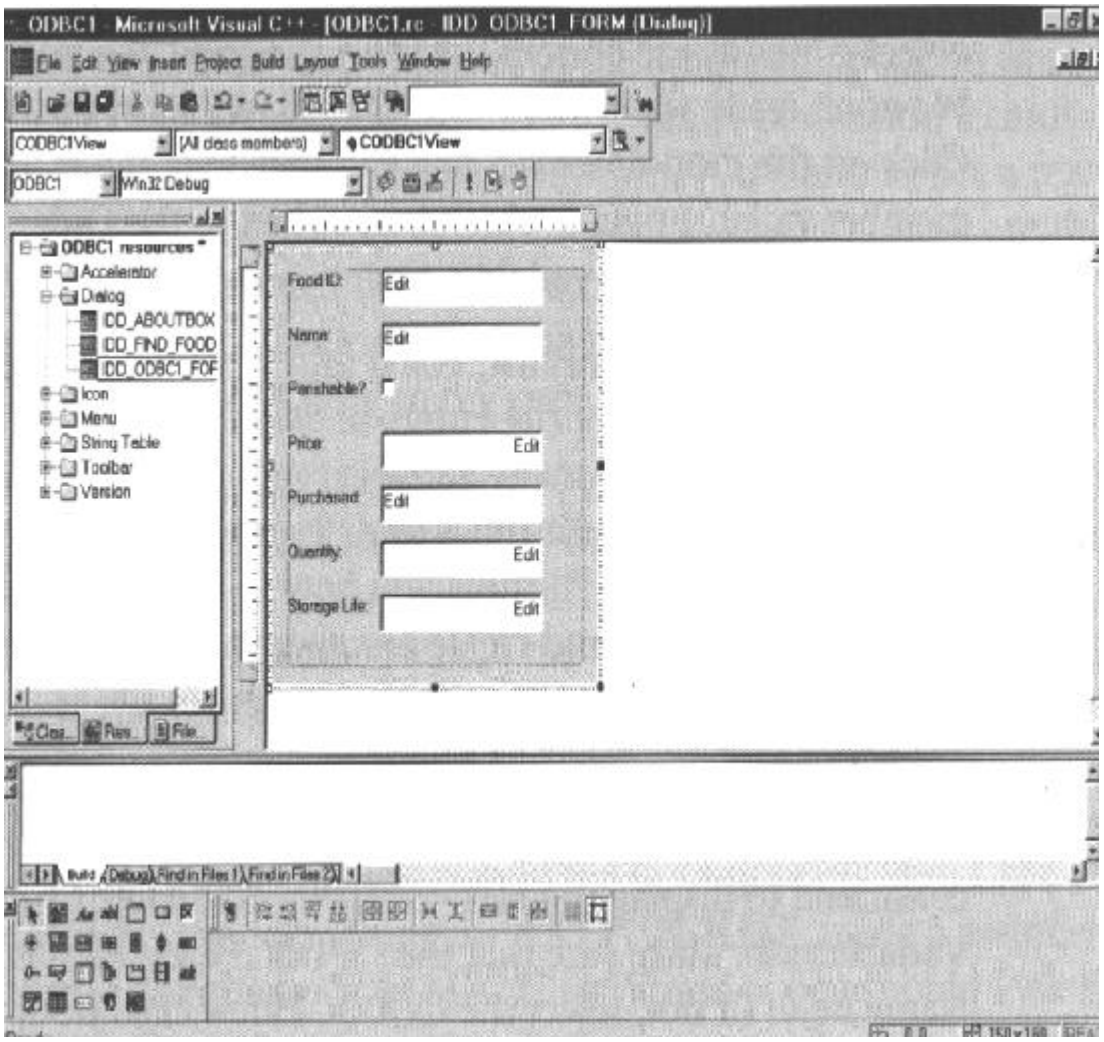
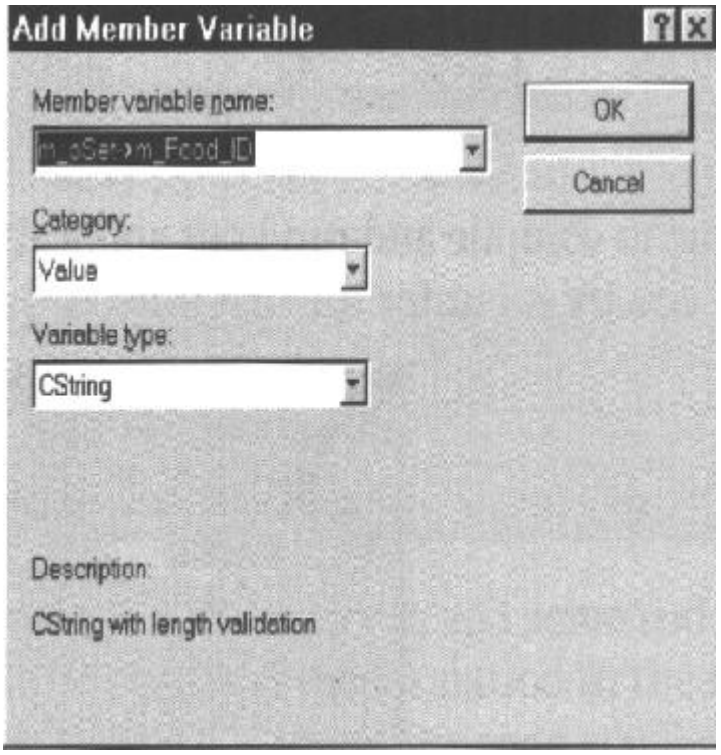


图 6.1　**我们要构造一个窗体来显示数据库中的数据**

现在我要说明为什么那些控件 ID **值非常重要。按住** CTRL **键并双击** IDC_FOOD_ID **控件，会看到一个如下图所示的** Add Member Variable **（添加成员变量）对话框。**



请注意，Visual C++**已经在** Member Variable Name**（成员变量名）字段中添加了一个值，但它不是我们需要的这个对话框中的正常值。通常你得到的变量以** m_**开头，而不再带其它杂七杂八的东西。此时，如果你熟悉代码的话，应该知道** m_pSet **是指向包含当前数据库记录的对象的指针。那个对象中的** m_Food_ID **变量包含** Foods**（食品）表中** Food_ID **字段的当前值。我们并不是真正想要那个** Food_ID **字段，而是想要** Orders**（订货）表中的** Food_ID **字段。单击下拉列表框，你会看到** m_pSet->m_Food_ID2 **成员变量名，它附属于** Orders**（订货）表。选择** Orders**（订货）表成员变量。**

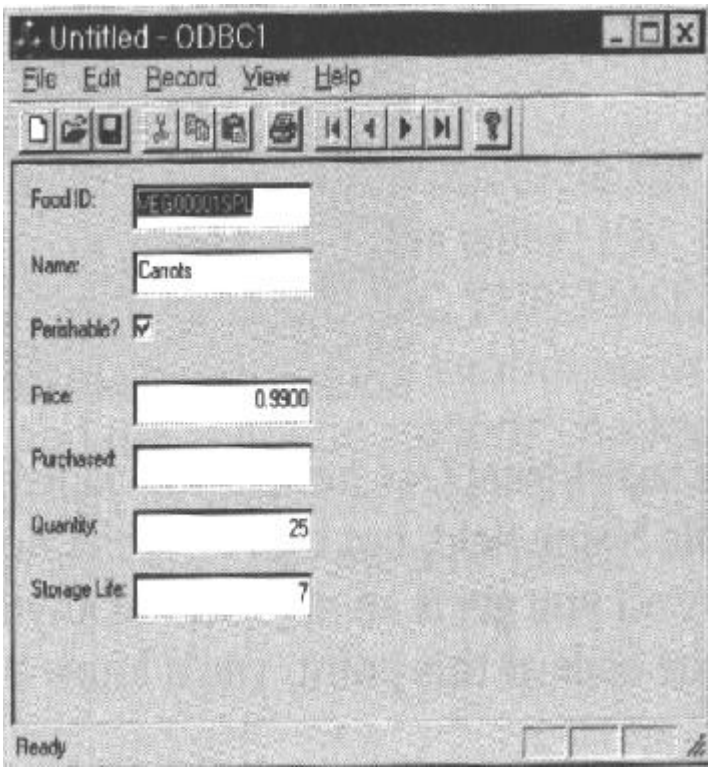
浏览这个列表，你会发现指向已构造查询中每一个字段的指针，Orders**（订货）表中的** Purchase**（采购）字段除外。**Visual C++**可以自动处理很多变量类型，**

包括大多数的成员。但是，它不能处理时间或日期，所以必须编写一些检索那些值的代码。稍后我会说明如何完成这个任务。现在，就像我们为 IDC_FOOD_ID 控件所做的那样，为剩下的控件创建成员变量。

一旦创建了所有的成员变量，请再复查一下，确保所做的一切都正确无误。用 View|ClassWizard（视图|类向导）命令显示 MFC ClassWizard 对话框。选择 Member Variables（成员变量）选项卡。确保 CODBC1View 类出现在 Class Name（类名）域中（如果不是这样，从下拉列表框中选择 CODBC1View 类）。成员变量列表应该和下图所示的列表相一致。

IDC_FOOD_ID2	CString	->m_Food_ID2
IDC_NAME	CString	->m_Name
IDC_PERISHABLE	BOOL	->m_Perishable
IDC_PRICE	CString	->m_Price
IDC_PURCHASE		
IDC_QUANTITY	long	->m_Quantity
IDC_STORAGE_LIFE	int	->m_Storage_Life

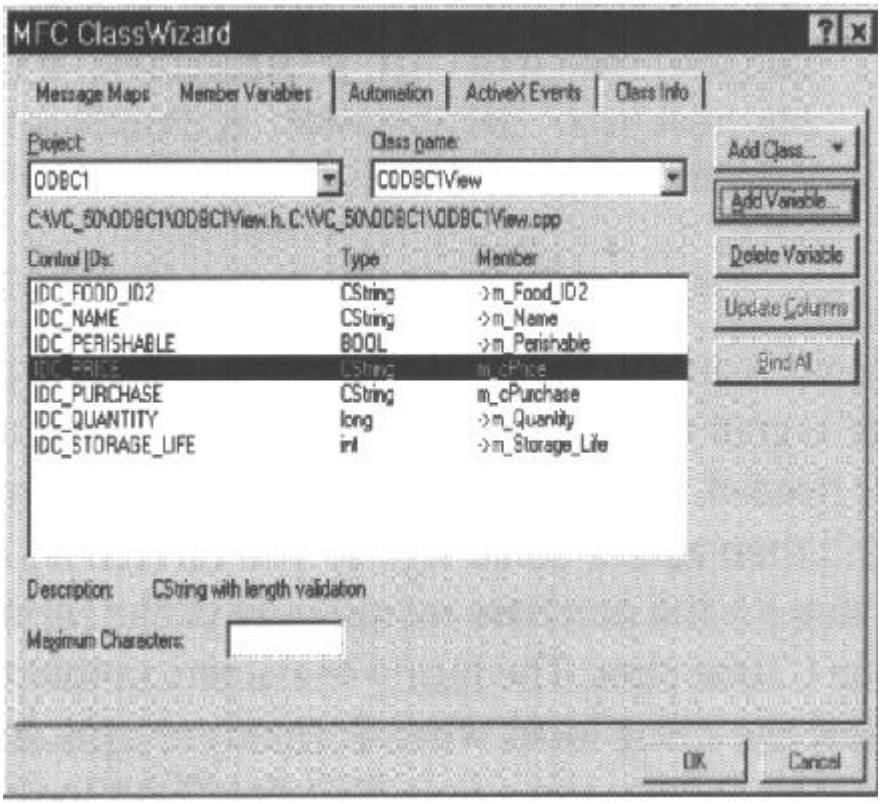
现在编译并运行示例应用程序。刚开始时应该能够看到下图所示的窗体。



运行应用程序后，就请试着在数据库中的记录间移动。你会注意到，这一部分程序运行得很好。程序还允许更改现有记录的值——至少为那些包含了值的字段。但是，你还会注意到一些不足。第一点你已经知道了——我们不在 Purchased（采购）字段中显示任何内容。第二个问题是，Price（价格）字段包含四个零位置，而不是通常和货币值有关的两个零位置。我们必须用代码解决这两个问题。

我们在做代码更改之前，要更改有关类成员变量的一些事。用 View|ClassWizard（视图|类向导）显示 MFC ClassWizard 对话框。象上一次所做的那样，选择 Member Variables（成员变量）选项卡和 CODBC1View 类。我们需要做的第一件事是，删除当前赋值给 IDC_PRICE 的变量。加亮它并单击

Delete Variable（删除变量）按钮。现在单击 Add Variable（添加变量）按钮，会看到一个 Add Member Variable（添加成员变量）对话框。为成员变量名输入 m_cPrice（Visual C++会自动供给 m_）。单击 OK 添加变量。为 IDC_PURCHASE 做相同的操作，这次用成员变量名 m_cPurchase。MFC ClassWizard 对话框看上去应该如下图所示。



现在我们有了一些变量，可以添加一些代码了。当然，我们想做的第一件事是显示数据库中的信息，这样在后面才能看到变化。程序列表 6.2 显示了需要添加到 CODBC1View 类的 DoDataExchange()函数的代码。要绝对确定，在函数的开始添加新的代码，而不是在函数的末尾添加。否则，会得到奇怪的结果。

程序列表 6.2

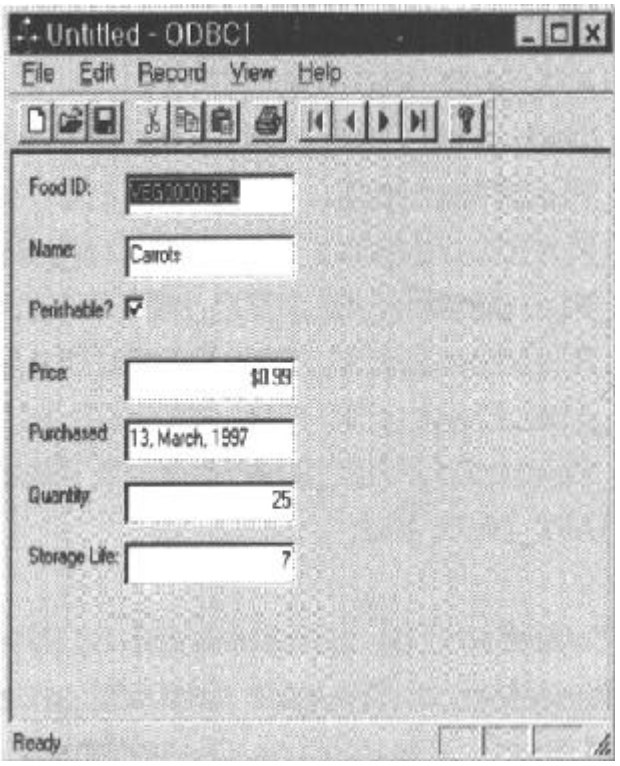
```
void CODBC1View::DoDataExchange(CDataExchange* pDX)
 {
 int                iStrLength;                // Length of the string.

 // Format the price for display by adding a $ and removing the
 // trailing zeroes.
 iStrLength = m_pSet->m_Price.GetLength();
 m_cPrice = "$" + m_pSet->m_Price.Left(iStrLength - 2);
 // Get the date value and convert it to a string.
 m_cPurchase = m_pSet->m_Purchase.Format("%d %B, %Y");

 CRecordView::DoDataExchange(pDX);
 //{{AFX_DATA_MAP(CODBC1View)
 DDX_Text(pDX, TDC_PRICE, m_cPrice);
 DDX_FieldText(pDX, IDC_FOOD_ID2, m_pSet->m_food_ID2, m_pSet);
 DDX_FieldText(pDX, IDC_NAME, m_pSet->m_Name, m_pSet);
 DDX_FieldCheck(pDX, IDC_PERISHABLE, m_pSet->m_Perishable, m_pSet);
 DDX_FieldText(pDX, IDC_QUANTITY, m_pSet->m_Quantity, m_pSet);
 DDX_FieldText(pDX, IDC_STORAGE_LIFE, m_pSet->mStorage_Life, m_pSet);
 DDX_Text(pDX, IDC_PURCHASE, m_cPurchase);
```

```
    //}}AFX_DATA_MAP
}
```

如你所见，解决显示问题相对比较容易。只要人工从 m_pSet **对象抓取值，然后按需要将其格式化。新代码从** Price**（价格）字段删除两个额外的零，然后添加一个美元符号，这样就可以告诉用户，它是一个货币数量。获取采购值要求使用** Ctime **类提供的** Format()**函数。字符串中有趣的字符**("%d %B，%Y")**确定** Format()**运行时字符串的外观。在这里，我想把日、月当作文本，还有** 4 **位数字的年。现在我们来看看程序是如何运行的。编译并运行应用程序，它应该如下图所示。**



尽管现在可以从数据库读取 Price**（价格）和** Purchased**（采购）字段了，但**

我们还不能更新它们（确保在添加代码以更新字段之前读取字段，这一点很重要；否则，如果应用程序不按预期运行，你就不知道代码在哪里出错了）。这一次，要添加一些代码到 DoDataExchange()**函数的末尾，因为我们想在和窗体上的控件交换数据之后再更新数据库。程序列表** 6.3 **显示了** DoDataExchange() **函数的最终形式。**

程序列表 6.3

```
void CODBC1View::DoDataExchange(CDataExchange* pDX)
 {
 int                iStrLength;          // Length of the string.
 int                iDay;                // Purchase day.
 int                iMonth;              // Purchase month.
 int                iYear;               // Purchase year.
 CTime              oTime;               // CTime object;

// Format the price for display by adding a $ and removing the
// trailing zeroes.
  iStrLength = m_pSet->m_Price.GetLength();
  m_cPrice = "$" + m_pSet->m_Price.Left(iStrLength - 2);

// Get the date value and convert it to a string.
  m_cPurchase =m_pSet- >m_Purchase.Format("%d %B, %Y");
```

```
CRecordView::DoDataExchange(pDX);
//{{AFX_DATA_MAP(CODBC1View)
DDX_Text(pDX, IDC_PRICE,m_cPrice);
DDX_FieldText(PDx,IDC_FOOD_ID2, m_pSet->m_Food_ID2, m_pSet);
DDX_FieldText(pDX, IDC_NAME, m_pSet->m_Name, m_pSet);
DDX_FieldCheck(pDX, IDC_PERISHABLE, m_pSet->m_Perishable, m_pSet);
DDX_FieldText(pDX, IDC_QUANTITY, m_pSet->m_Quantity, m_pSet);
DDX_FieldText(pDX, IDC_STORAGE_LIFE, m_pSet->mStorage_Life, m_pSet);
DDX_Text(pDX, IDC_PURCHASE, m_cPurchase);
//}}AFX_DATA_MAP

// Transfer the information to the m_Price variable.
// Remember to remove the dollar sign, if necessary, first.
iStrLength = m_cPrice.GetLength();
if (m_cPrice.Find("$") != -1)
{
        m_pSet->m_Price = m_cPrice.Right(iStrLength - 1);
}
else
{
```

```
        m_pSet->m_Price = m_cPrice;
}

// Get the updated year.
iYeat = atoi(m_cPurchase.Right(4));

// Get the updated day.
iDay = atoi(m_cPurchase..Left(m_cPurchase.Find(" ") + 1));

// Get the updated month.
if (m_cPurchase.Find("January") != -1)
        iMonth = 1;
if (m_cPurchase.Find("February") != -1)
        iMonth =2;
if (m_cPurchase.Find("March") != -1)
        iMonth =3;
if (m_cPurchase.Find("April") != -1)
        iMonth =4;
if (m_cPurchase.Find("May") != -1)
        iMonth =5;
if (m_cPurchase.Find("June") != -1)
```

```
        iMonth =6;
  if (m_cPurchase.Find("July") != -1)
        iMonth =7;
  if (m_cPurchase.Find("August") != -1)
        iMonth =8;
  if (m_cPurchase.Find("September") != -1)
        iMonth =9;
  if (m_cPurchase.Find("October") != -1)
        iMonth =10;
  if (m_cPurchase.Find("November") != -1)
        iMonth =11;
  if (m_cPurchase.Find("December") != -1)
        iMonth =12;

// Transfer the date information to m_Purchase.
oTime = CTime(iYear iMonth, iDay, 0, 0,0);
m_pSet->m_Purchase = oTime;
}
```

上述程序看上去代码可不少，但是你可以相当容易地把它们分成几部分。我们要做的第一件事是添加一些新的变量来处理 Purchase **字段。还有几个变量表示日、月和年。我们还需要一个** Ctime **对象，它允许我们创建一个要放回到**

数据库中的日期。

请留意，Price 字段的更新需要进行简单的变换。但是，如果 Price（价格）字段包含了美元符号的话，就不能使用它，所以代码用 IF 语句来执行两个层次的处理：在 Price（价格）字段中，一个有美元符号，而另一个没有。请注意，这是第一次使用 Cstring 对象的 Find()函数来确定是否存在美元符号。你会在代码中相当多地使用 Find()函数，所以在寻找解决问题的办法时记住这个方法是十分方便的。

Purchase 字段在应用程序中是 Cstring 类型，除非我们花一点时间进行分析，否则它不会轻易地转换成 Ctime 对象。我们的代码假定用户总是以这样的格式输入日期：1 December 1998。可惜的是，创建完全成熟的数据库应用程序时，你可能需要考虑比我们的示例程序更多的偶然性。我们故意使示例的代码很简单。用 Cstring 提供的函数和简单地调用 atoi()函数来转换年和日是非常简单的。月份的表达倒是有点挑战性的味道，它需要一个个地实际编写在代码中。一系列的 IF 语句都是为了处理月份。最后，我们采用日期的三个整数表示方法，并把它转换到 Ctime 对象中。但我们有了 Ctime 对象后，数据库更新操作也就只需要一次简单的传输了。

如果现在运行应用程序，可以检索并编辑我们原始查询中的全部字段。正如你所看到的，在 Access 中只需要 4，5 分钟的窗体创建过程，在 Visual C++中却需要花上数小时的时间来实现。在 Visual C++赋予你众多额外灵活性的同时，就要付出编写额外代码的代价，对这一点应该有个清醒的认识。

6.2 操纵数据库的内容

要真正编写数据库应用程序的话，前面的示例确实派不上大用场。甚至，一个实用程序的功能都不只是限于阅读数据库的内容，而这正是目前我们所能全部做到的。例如，我们甚至还没有在数据库中执行过添加新记录或者查找数据的方法。可以说，这两种特征是任何功能数据库应用程序甚至是实用程序的一部分。在本章节中，我们要向示例数据库中添加这两种功能。

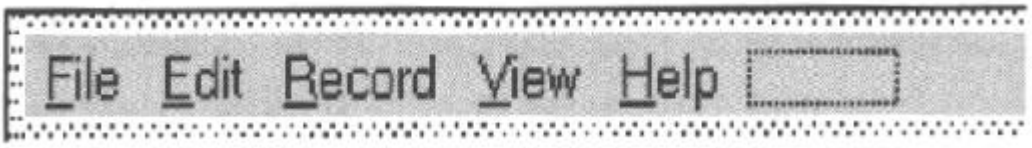
向表中添加记录

只要在设计阶段做出了正确的选择，那么向表中添加新记录的实际过程就不会困难。在大多数情况下，DBMS 会为你处理一切。你只需要以正确的方式提供正确的信息。例如，你可能想知道，为什么在本章上一节设计的窗体中，我选择使用 Orders 表的 Food_ID 字段，而不是 Foods 表的相应字段。一旦你考虑到了这一点，原因相对就简单了。使用表的 Food_ID 字段允许 DBMS 做出决定。如果在附加记录以后输入一个现有的 Food_ID 值，DBMS 会简单地把新记录添加到 Orders 表中。但是，如果输入一个新的 Food_ID 值，DBMS 就会把新记录添加到 Foods 和 Orders 两个表中。如果使用 Foods 表的 Food_ID 字段，就不可能有这个决定——DBMS 每次都会指示把新记录添加到两个表中。如果用户输入现有 Food_ID 值的话，会引起主键值重复的结果，并最终导致新记录被拒绝。

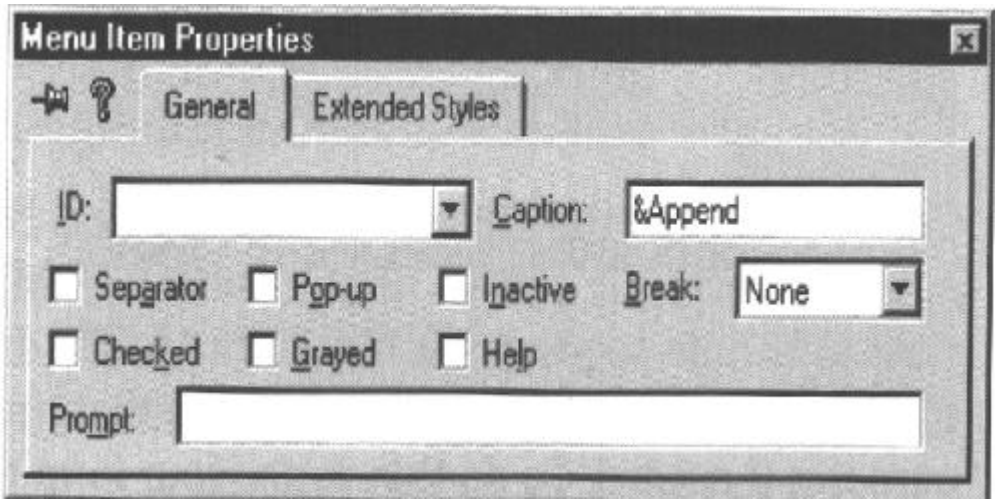
还有其它要考虑的原则。例如，你的应用程序处理数据有效性了吗？或者 DBMS 会为你处理数据有效性吗？在多数情况下，如果处理数据有效性的话，你可以提供一个更为友好的用户界面，但这样做也意味着应用程序会增大。我们的应用程序并没有执行任何数据有效性检查工作，原因有两个：第一，我想让这个示例足够简单，你可以看到所发生的一切。第二，它是一个实用型的程序，即大多数使用这个应用程序的人是像你一样的专业人员。

在向应用程序中添加任何代码之前，必须执行一些设置。要做的第一件事是，添加一个增加记录的菜单项。下面的过程说明了如何完成这项任务（我们已经在第 3 章中学习了添加菜单项的方法，所以这里就不再讨论每一个详细步骤了）。

1. 在 Visual C++中选择 ResourceView（资源视图）。双击 Menu（菜单），然后双击 IDR_MAINFRAME 会看到如下图如示的菜单栏。



2. 单击 Record（记录）菜单，然后单击菜单底部的空白条目。输入&Append（添加），会看到如下图所示的 Menu Item Properties（菜单项属性）对话框。



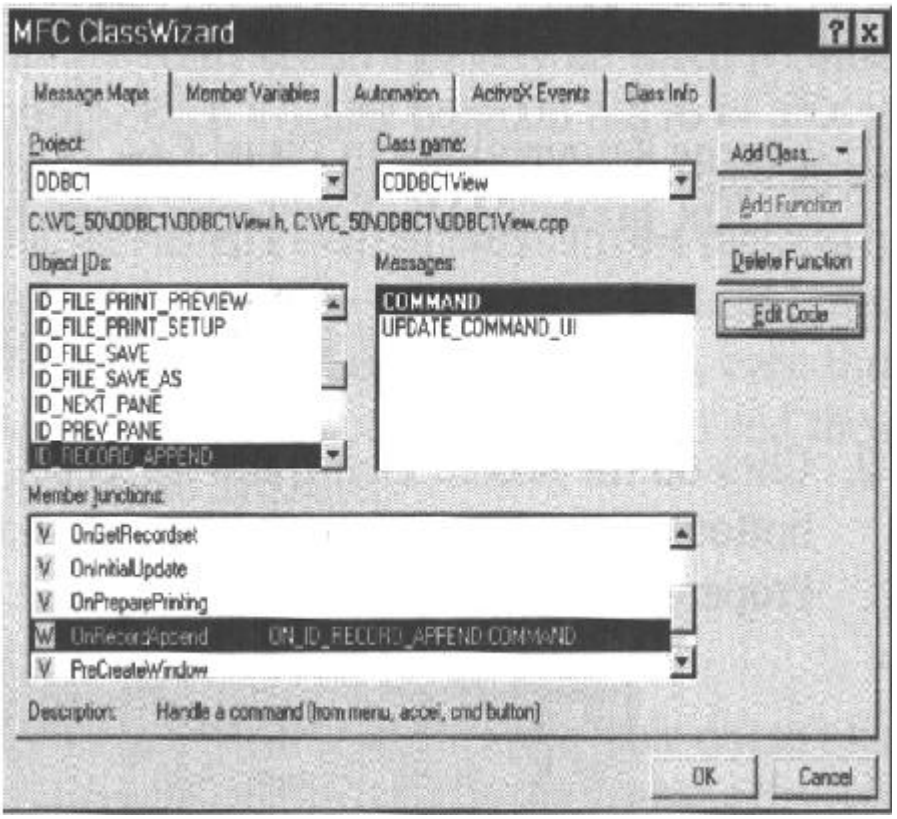
3. **在** Prompt **（提示）域中输入** Append a new record to the table.\nAppend Record。

4. **单击** Menu Item Properties **（菜单项属性）对话框中的** Close **（关闭）框。** Visual C++**自动把** ID_RECORD_APPEND ID **的值赋给新的菜单项（双击的话，可以看到实际完成的情况）。**

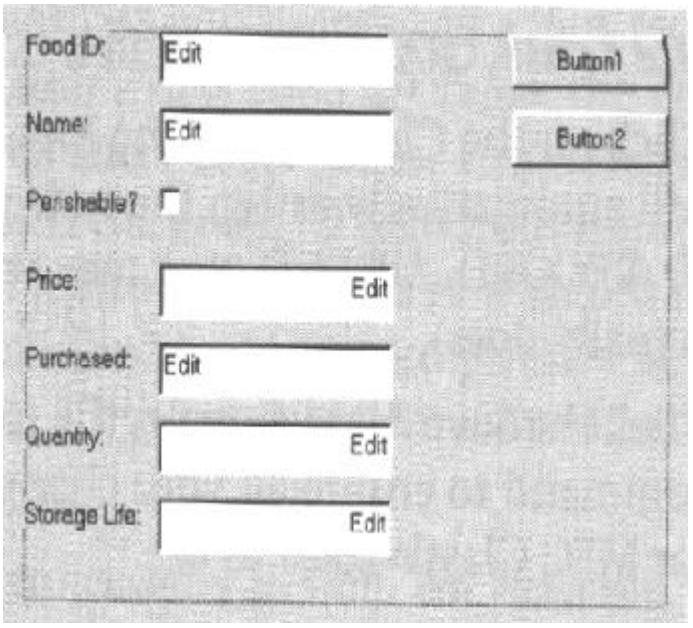
5. **单击工具条上的** Save All **（全部保存）按钮或使用** File|Save All **（文件|全部保存）命令，确定已保存了全部的更改并进行了正常的登记，以便** MFC ClassWizard **能够使用。**

现在我们有了可供使用的菜单项，下面要在 ClassWizard **中添加一些条目，以使其发挥作用。右击菜单条目，从上下文菜单中选择** ClassWizard **（类向导）。在** MFC ClassWizard **对话框中，应该看到** Object IDs **列表中已选择了** ID_RECORD_APPEND **条目（否则请选择它）。在** Class Name **（类名）下拉列表框中选择** CODBC1View**——这个类允许你查看数据库的内容，所以还要在这里为添加记录增加一些代码。加亮** Messages **（消息）列表中的** COMMAND **（命**

令）条目，然后单击 Add Function（添加函数），你会看到 Add Member Function（添加成员函数）对话框。单击 OK 接受缺省的函数名。MFC ClassWizard 对话框看上去如下图所示。



单击 OK 关闭 MFC ClassWizard 对话框。现在我们要修改先前创建的窗体(IDD_ODBC1_FORM)。有几种处理向数据库中添加新记录的方式，但是我已经选择了添加按钮——实际上这是个一般按钮，如果需要的话，也可以把这个按钮用于其它目的。这时，把窗体大小调整为 220×160，然后添加两个按钮。窗体的外观如下图所示。

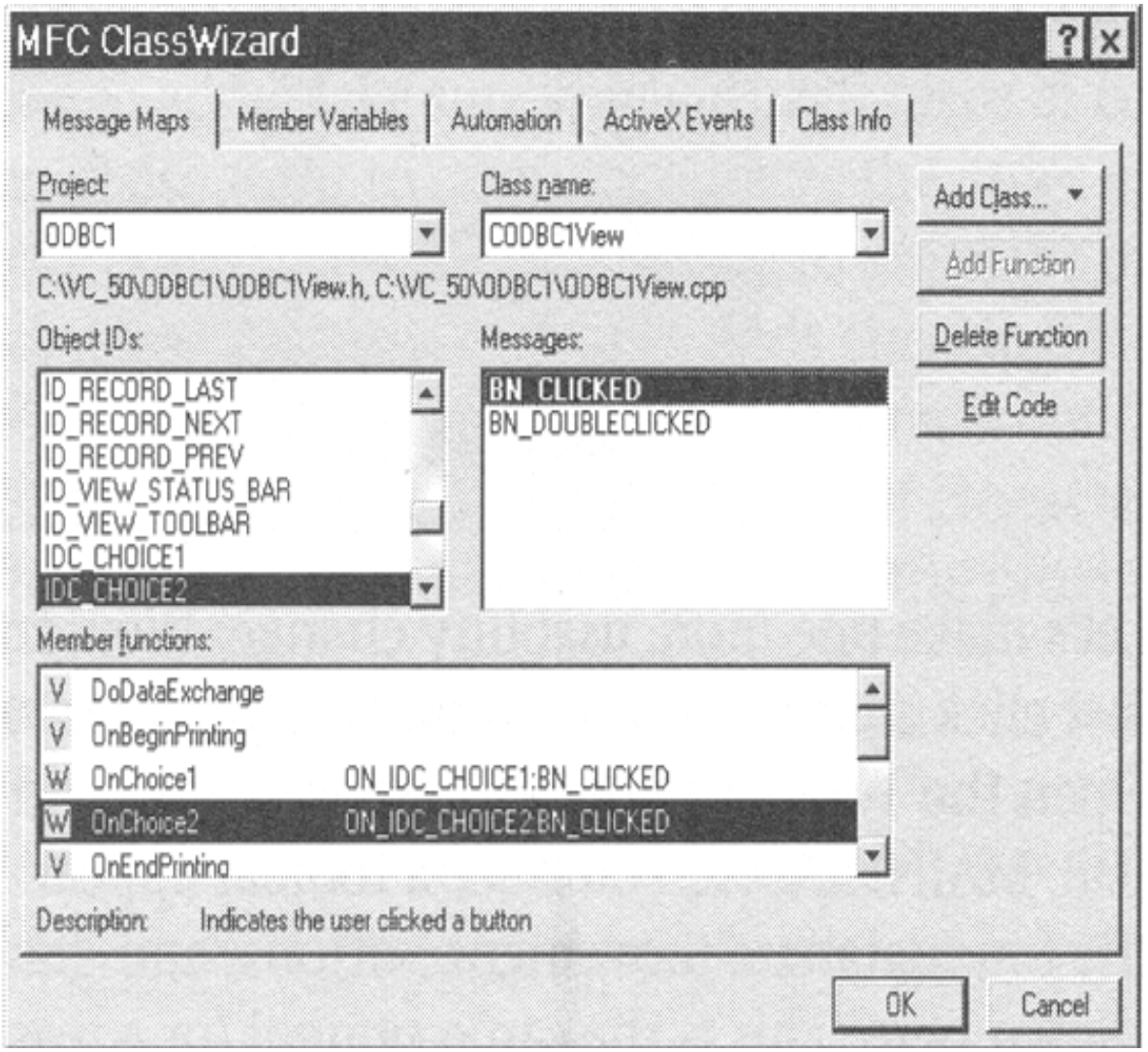


我们始终不想让按钮处于可见状态（因此可访问），所以需要修改几个属性。右击第一个按钮，然后从上下文菜单中选择 Properties（属性），打开 Properties（属性）对话框。清除 Visible（可见）属性的选中标记，把 ID 修改为 IDC_CHOICE1。对第二个按钮执行相同的操作，把它的 ID 修改为 IDC_CHOICE2。

注 如果想在对话框或窗口刚显示时隐藏某个控件，那么请不选中该控件的 Visible 属性。

添加一些函数来处理用户在按钮上的单击操作，这一点也很重要。你只要右击第一个按钮，从上下文菜单中选择 ClassWizard（类向导）选项。选择 Message Maps（消息映射）选项卡。在 Class Name（类名）字段中选择 CODBC1View。在 Object IDs（对象 ID）列表中找到 IDC_CHOICE1 条目并加亮它。现在加亮 Messages（消息）字段中的 BN_CLICKED 条目，然后单击 Add Function（添加

函数）创建处理用户单击操作的函数框架。要为 IDC_CHOICE2 Object ID 条目进行相同的操作。下面是 Member Functions（成员函数）列表的外观。

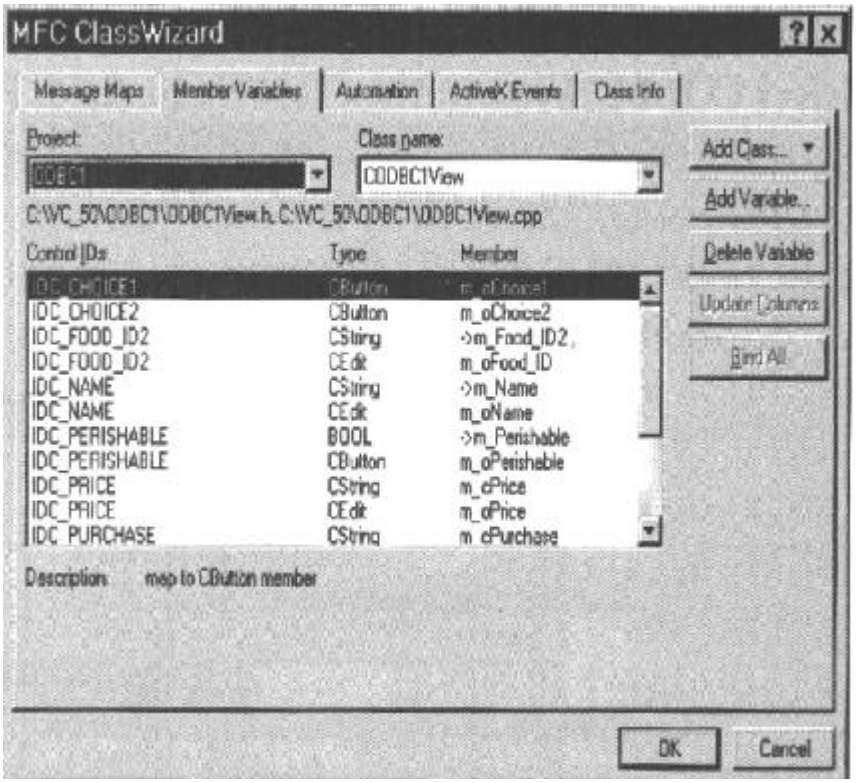


单击 OK 关闭 MFC ClassWizard 对话框。我们以前已经为窗体上的每一个数据项字段都添加了一些成员变量，那些成员变量允许程序和数据库交换数据。但是，现在我们需要直接访问那些数据项字段，所以要再创建一些成员变量。

注 Visual C++允许你把多个成员变量赋给单个对象。

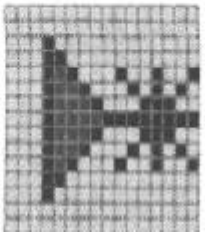
按住 CTRL 并双击窗体上的 IDC_FOOD_ID 控件。会看到 Add Member Variable（添加成员变量）对话框。在 Member Variable Name（成员变量名）域中输入 m_oFood_ID，然后在 Category（分类）域中选择 Control（控件）。单击 OK 完成这个过程。刚刚完成的是创建一个成员变量，它可以直接访问窗体上的控件。用下面的成员变量名为其它数据项控件执行相同的任务：IDC_NAME(m_oName)，IDC_PERISHABLE(m_oPerishable) IDC_PRICE(m_oPrice)，IDC_PURCHASE(m_oPurchase)，IDC_QUANTITY(m_oQuantity)和 IDC_STORAGE_LIFE(m_oStorage_Life)。

我们来检查一下添加新变量的结果（这个想法不错，因为你要确保可以实际访问它们）。右击任何地方，然后从上下文菜单选择 ClassWizard（类向导）。选择 MFC ClassWizard 对话框的 Member Variables（成员变量）选项卡。应该在窗体上看到每个数据项控件的两个变量，如下图所示。

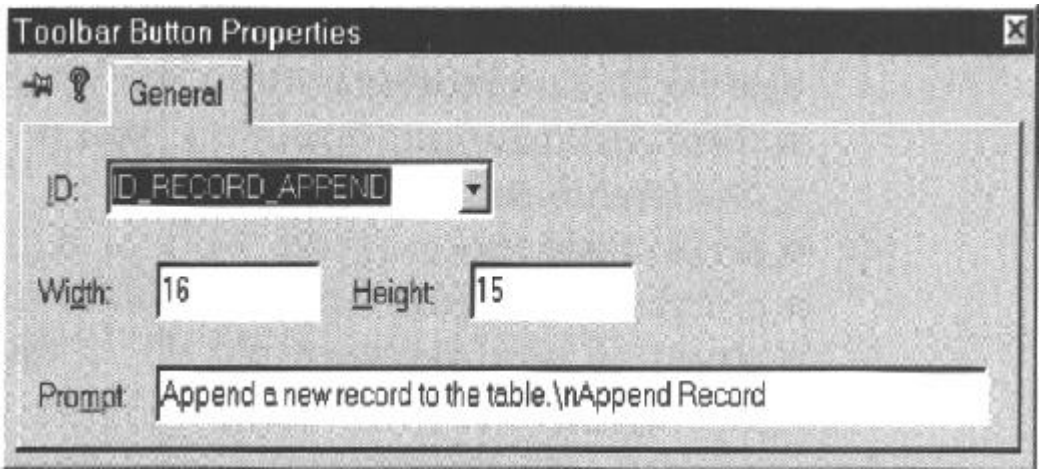


我们再做一点提供程序可用性的修改。如果每次用户只是单击工具条上的按钮而不使用菜单的话，结果不是更好吗？打开 ResourceView **（资源视图）中的** Toolbar **（工具条）文件夹，然后双击** IDR_MAINFRAME**。你会看到系统显示一个工具条。在四个数据库移动按钮后面添加一个新按钮，并在上面画一个如下图所示的符号。这是和** Access **中** New Record **（新记录）按钮图案相同的符号（或者是合理的模仿符号）。**





要这个按钮发挥作用并不太困难。双击工具条上新的 Append Record（添加记录）按钮（不是刚才你在绘图区所画的按钮，而是在工具条上的按钮），你就会看到 Toolbar Button Properties（工具条按钮属性）对话框。从 ID 下拉列表框中选择 ID_RECORD_APPEND。这把按钮和菜单命令联系起来。Prompt（提示）字段应该自动以正确的文本值填充，如下图所示。



单击 Close（关闭）框完成本次操作。这些就是把按钮添加到工具条上并使其发挥作用的全部操作步骤。

现在向这一部分示例中添加一些代码。程序列表 6.4 说明了需要添加的代码。一定要把每一段代码都添加到正确的函数中。因为一些操作必须以特定的

顺序进行，所以还要仔细地观察任务完成的顺序。最后，这个代码很明显不做任何类型的边界检查，所以在测试它时必须仔细地输入。Food ID 字段一定要包含足够多的类型正确的字符。如果犯了某个错误而程序不正常恢复的话，请使用 Access 手工编辑表。

程序列表 6.4

```
void CODBC1View::OnRecordAppend()
 {
      // See if the database was opened in read-only mode. If so,
      // display a message, and then exit.
      if (m_pSet->CanAppend() == 0)
            MessageBox("Cannot Append Records",
                  "Database Opening Error",
                  MB_OK | MB_ICONERROR);
     else
    {
          // Set the option button captions.
          m_oChoice1.SetWindowText("Submit Data");
          m_oChoice2.SetWindowText("Cancel");

        // Display the option buttons.
        m_oChoice1.Showwindow(SW_SHOW);
```

```
            m_oChoice2.Showwindow(SW_SHOW);

            // Create a blank record for the user to edit.
            m_oFood_ID.SetWindowText(" ");
            m_oName.SetWindowText(" ");
            m_oPerishable.SetCheck(0);
            m_oPrice, SetWindowText(" ");
            m_oPurchase.SetWindowText(" ");
            m_oQuantity.SetWindowText(" ");
            m_oStorage_Life.SetWindowText(" ");
        }
    }

void CODBC1View::OnChoice1()
{
    Cstring     cValue;         // Current value of edit field.
    int         iCheck;         // Current check value of Perishable field.
    int         iStrLength;     // Length of the string.
    int         iQuantity;      // Value of Quantity field;
    int         iStorage;       // Value of Storage field;
    int         iDay;           // Purchase day.
```

```
    int             iMonth;                 // Purchase month.
    int             iYear;              // Purchase year.
    CTime           oTime;              // CTime object;
    BOOL            lNew = TRUE;        // New Food ID?

    // Initialize the search value.
    m_oFood_ID.GetWindowText(cValue);

  // Go to the beginning of the query and search
  // for the Food ID entered by the user.
  m_pSet->MoveFirst();
  while (!m_pSet->IsEOF()   ^ !lNew)
 {
        // Check if the value is equal.
        if (m_pSet->m_Food_ID    == cValue)
              lNew = FALSE;

        // Go to the next record.
        m_pSet->MoveNext();
}
```

```
// Add a new record.
m_pSet->AddNew();

// Save the new data values into the record.
m_oFood_ID.GetWindowText(cValue);
m_pSet->m_Food_ID2 = cValue;
m_oFood_ID.GetWindowText(cValue);
m_pSet->m_Name = cValue;

// If this is a new record, place the Food ID in the
// Foods table field as well.
if (lNew )
{
	m_oFood_ID.GetWindowText(cValue);
	m_pSet->m_Food_ID = cValue;
}

//Convert Perishable control value from int to BOOL.
iCheck = m_oPerishable.GetCheck();
if (iCheck == 0)
	m_pSet->m-Perishable = FALSE;
```

```
  else
        m_pSet->m-Perishable = TRUE;

// Convert and then save the Quantity and Storage fields.
  m_oQuantity.GetWindowText(cValue);
  iQuantity = atoi(cValue);
  m_oStorage_Life.GetWindowText(cValue);
  iStorage = atoi(cValue);
  m_pSet->m_Quantity = iQuantity;
  m_pSet->m_Storage_Life = iStorage;

  // Transfer the information to the m_Price variable.
  // Remember to remove the dollar sign, if necessary, first.
  iStrLength = m_cPrice.GetLength();
  if (m_cPrice.Find("$") != -1)
  {
        m_pSet->m_Price = m_cPrice.Right(iStrLength - 1);
  }
  else
  {
        m_pSet->m_Price = m_cPrice;
```

```
}

// Get the updated year.
iYear = atoi(m_cPurchase.Right(4));

// Get the updated day.
iDay = atoi(m_cPurchase.Left(m_cPurchase.Find(" ") + 1)),
// Get the updated month.
if (m_cPurchase.find("January") != -1)
        iMonth = 1;
if (m_cPurchase.Find("February") != -1)
        iMonth = 2;
if (m_cPurchase.Find("March") != -1)
        iMonth = 3;
if (m_cPurchase.Find("April") != -1)
        iMonth = 4;
if (m_cPurchase.Find("May") != -1)
        iMonth = 5;
if (m_cPurchase.Find("June") != -1)
        iMonth = 6;
if (m_cPurchase.Find("July") != -1)
```

```
        iMonth = 7;
if (m_cPurchase.Find("August") != -1)
        iMonth = 8;
if (m_cPurchase.Find("September") != -1)
        iMonth = 9;
if (m_cPurchase.Find("October") != -1)
        iMonth = 10;
if (m_cPurchase.Find("November") != -1)
        iMonth = 11;
if (m_cPurchase.Find("December") != -1)
        iMonth = 12;

// Transfer the date information to m_Purchase.
oTime = CTime(iYear, iMonth, iDay, 0, 0, 0);
m_pSet->m_Purchase = oTime;

// Once the user completes the record, update the database.
m_pSet->Update();

// Go to the last record so that the user can see the addition.
m_pSet->MoveLast();
```

```
    // Hide the option buttons.
    m_oChoice1.ShowWindow(SW_HIDE);
    m_oChoice2.ShowWindow(SW_HIDE);
}

void CODBC1View::OnChoice2()
{
    int          iStrLength;          // Length of the string.
    char*        cQuantity = "Empty1";  // Text value of Quantity field.
    char*        cStorage = "Empty2";   // Text value of Storage Life field.

    // Hide the option buttons.
    m_oChoice1.showWindow(SW_HIDE);
    m_oChoice2.showWindow(SW_HIDE);

    // Restore the display data.
    m_oFood_ID.SetWindowText(m_pSet->m_Food_ID2);
    m_oName.SetWindowText(m_pSet->m_Name);
    m_oPerishable.SetCheck(m_pSet->m_Perishable);
```

```
    // Process the Quantity and Storage Life fields.
    itoa(m_pSet->m_Quantity,cQuantity, 10);
    itoa(m_pSet->m_Storage_Life, cStorage, 10);
    m_oQuantity.SetWindowText(cQuantity);
    m_oStorage_Life.SetWindowText(cStorage);

    // Format the price for display by adding a $--the trailing
    // zeroes were removed during the data exchange.
    iStrLength = m_pSet->m_Price.GetLength();
    m_oPrice.SetWindowText("$" + m_pSet->m_Price);

    // Get the date value and convert it to a string.
    m_oPurchase.SetWindowText(m_pSet->m_Purchase.Format("%d %B, %Y"));
}
```

看上去上述代码可真不少，但如果把它们分开来看的话，也就不难理解了。我们从 OnRecordAppend()**函数开始。应该做的第一件事是，检查数据库是否允许你添加记录。可以用** CRecordset **类的** CanAppend()**函数来完成这个工作。如果可以添加记录，这个函数返回** TRUE**。如果应用程序不能添加记录，它就显示一条出错消息，并像示例代码中那样退出。此时，你可以以只读方式之外的方式重新打开数据库。**

OnRecordAppend()**函数的第二部分完成三个任务。第一，我们用与** CWnd

类相关联的 SetWindowsText()给添加到窗体的两个通用按钮赋上一个标题。类总是继承其导出的基类的特征，认识这一点很重要。如果你查阅一下 CButton 类的文档，就会发现不能像 Visual Basic 语言那样设置 Caption（标题）属性，所以完成这项工作需要一个像 SetWindowsText()这样的函数（尽管函数名好像与其功能不相配）。第二，我们用 CWnd 类的 ShowWindow()函数使按钮可见。同样，这一点可能与直觉有些不符，但它确实工作得很好。第三，我们要创建空白编辑框让用户编辑。你还可以提供建议值或者其它适合的值来填充 CEdit 控件。

我们暂时先跳过 OnChoice1()函数（用户单击 Submit（提交）按钮时将调用这个函数），看看如果用户单击 Cancel（取消）（它调用 OnChoice2()函数）时会发生什么。OnChoice2()函数有两部分。第一部分应该很熟悉了。这一次我们不是让 CEdit 控件可见，而是把它隐藏起来。

第二部分检索当前的查询值，并把它放回到窗体上的 CEdit 控件中。对用户来说，就好像是你刚刚恢复了记录，而实际完成的是恢复窗体中的值。请注意，我们必须用四种不同的技术来恢复前面的值。第一种技术是直接数据传输。第二种技术是把数据库提供的数值转换成字符串，然后把字符串放入窗体中。第三种技术重复我们在上一节所做的工作——它只是格式化现有的字符串，所以实际显示成货币值。第四种技术使用 Format()函数把日期转换成字符串。

现在我们来看一看，如果用户填充窗体并单击 Submit Data（提交数据）按钮的话，会发生什么。请记住，为了演示方便，这段代码被简化了——可以添加一些代码来验证每个字段都包含了数据库可能实际使用的值。OnChoice1()函

数需要做的第一件事是，看看我们是否正在向 Foods（食品）表中的现有 Food ID 添加新条目，或者是否我们正在添加全新的 Food ID。这个差别是很重要的，你只会在几个段落中看到。查找这个信息的过程分成如下四个步骤。

1. 获得用户提供的 Food ID 值。在本例的情况下，我们用 GetWindowText() 函数来完成。

2. 使用 CRecordset 类的 MoveFirst()函数移动到查询的开始位置。

3. 将查询的 Food ID 字段中的每一个值与用户提供的值进行比较。请注意，我们用 MoveNext()函数在记录间移动。

4. 如果找到了一个匹配值，那么把 1New 设置成 FALSE 并退出搜索循环。

一旦你决定了要添加哪种类型的记录，你就需要使用 AddNew()函数向数据库中添加一个空白记录。此时，这个空白记录只是保存在内存中。如果用户的机器出现电源故障，那么数据库中便不会有空白记录。此时，空白记录只是作为一个模板（如果你喜欢，也可以是空白窗体）来使用，以保存用户提供的数据值。

现在向我们的空白记录中添加数据。请注意，从窗体获取数据的过程不像最初想像得那样简单。最简单的传输过程是从 CEdit 控件到查询中的字符串字段之间的传输。这个过程分成两个步骤。要做的第一件事是，用 GetWindowText() 函数把 CEdit 控件中当前的文本放入一个临时变量。填充了临时变量后，就可以把信息传输到数据库字段中了。

复选框很有意思。Visual C++支持的复选框可以包含三个值中的一个：选中，未选中或者不确定。结果，CButton 类的 GetCheck()函数返回一个整数，而不是

Boolean（布尔）值。你需要把这个整数值转换成 Access 使用的 Boolean（布尔）值。

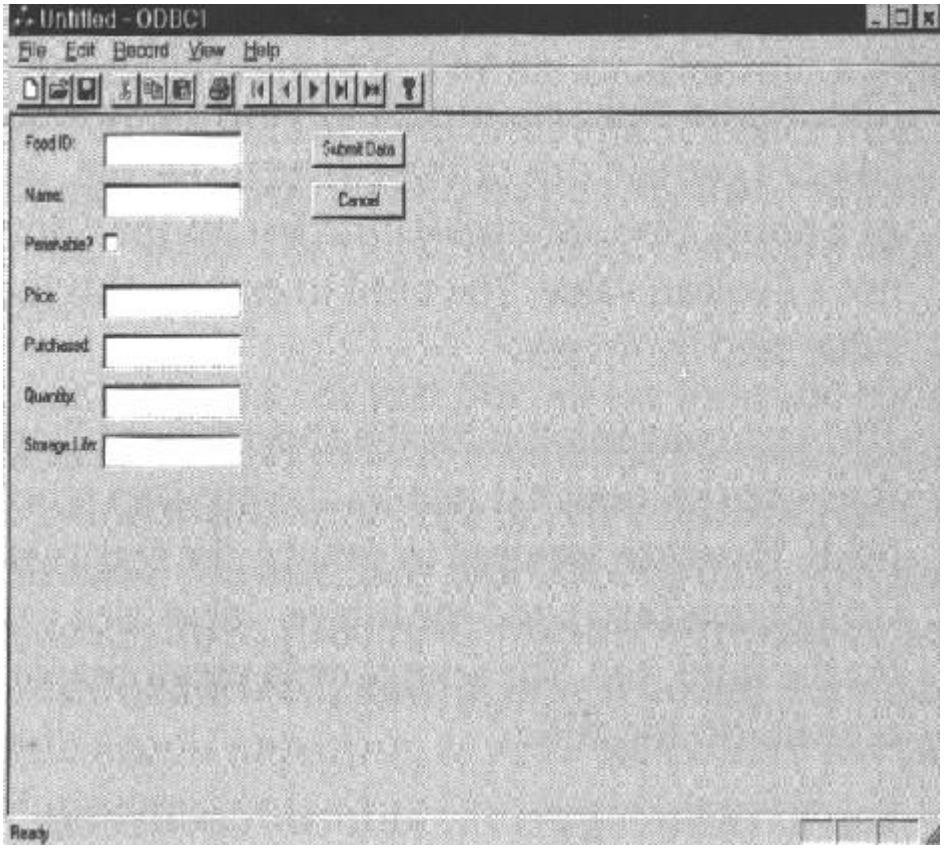
下一个转换发生在 Quantity 和 Storage Life 字段中。这个过程分成三个步骤，模仿了我们为文本字段使用的二步骤过程。但是，我们要把用 GetWindowText() 获得的文本值转换成字段使用的整数值——这算是第三步。示例代码用简单的 atoi()函数来完成这项任务。

你应该见过下面的两个转换。它们都以相对未变的形式出现在 DoDataExchange()函数中。你仍然需要得到窗体上控件提供的 Purchased（购买）字段的文本值，并把它转换成日期。这一点没有任何变化。同样地，要从 Price（价格）字段删除美元符号。否则，Access 不会接受字段值。

现在我们有了填满了新数据值的空白窗体，可以用 Update()函数来更新数据库了。这是数据库唯一真正脆弱的时刻。如果在更新的很短暂的时间内出现了电源故障，可能会得到不可预知的后果。必须说，虽然这种方法不安全，但它确实要比长时间打开记录的老技术的数据库好。

OnChoice1()函数与数据库相关的最后一个动作是移动到最后一个条记录。你必须完成这个操作，这样用户才能实际看到新的记录。否则，用户可能会认为记录没有放到数据库中，并试图再添加一个。请注意，OnChoice1()函数以隐藏两个通用按钮来结束这个函数。除非用户实际上有新的记录要编辑，否则他们不能访问这些按钮，这一点很重要。所以，要尽可能地把它们隐藏起来。

那么，空白记录看上去是什么样子呢？下图是我们的示例程序显示的空白记录。

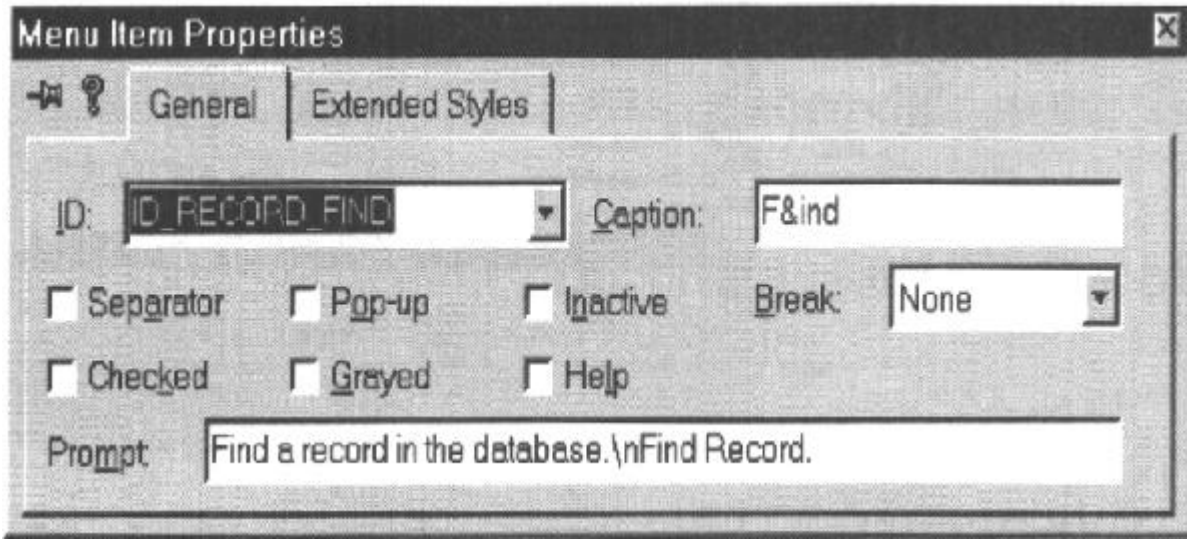


查找数据库中的数据

不管应用程序看上去有多好，如果用户找不到他们所需要的东西，那么它也不会有太大的价值。构造出使用户在数据库中移动的最佳方式一直是众说纷纭的主题。有一些书，它们整章的主题都是关于以尽可能有效的方式查找所需要的东西。我们应该花很多的时间来学习搜索技术，但最后也并不会为你提供太多的帮助。使用 ODBC 时，有一种简单的查询方式；它也许不是最快或者最有效，但肯定能找到你所需要的东西。我们要在本节中集中讨论这项技术。

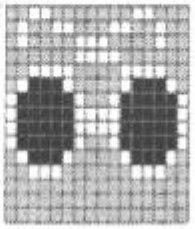
很明显，在我们进一步深入探讨之前，还要再在菜单和工具条上花点时间。

再次打开 IDR_MAINFRAME 菜单，这一次我们准备在 Record（记录）菜单下添加 Find（查找）菜单项。选择最后的空白并输入 F&ind（注意，已经为 First Record（第一个记录）选项在 F 下加了下划线）。还要在 Prompt（提示）域中输入下述提示：Find a record in the database.\nFind Record。此时的 Menu Item Properties（菜单项属性）对话框应该如下图所示。



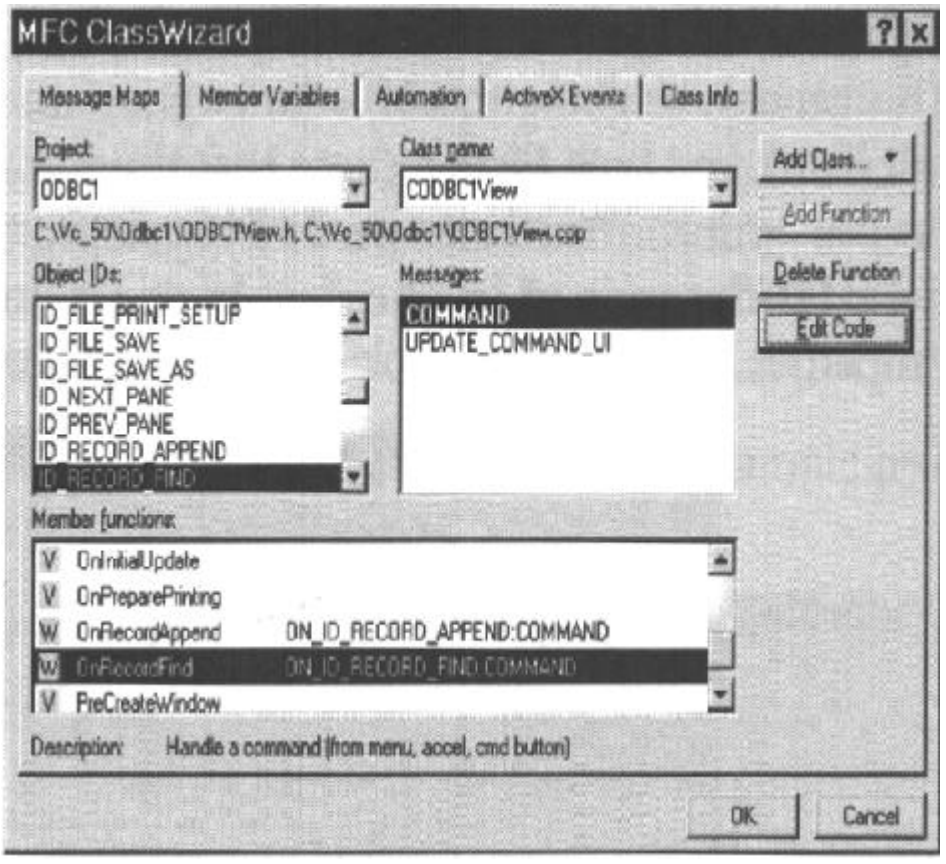
单击 Close（关闭）框完成新菜单项的创建工作。打开 IDR_MAINFRAME 工具条。下面是我添加的新的 Find（查找）按钮，你当然也可以使用任何一种想要的按钮（用不着奇怪——它确实是一副眼镜）。



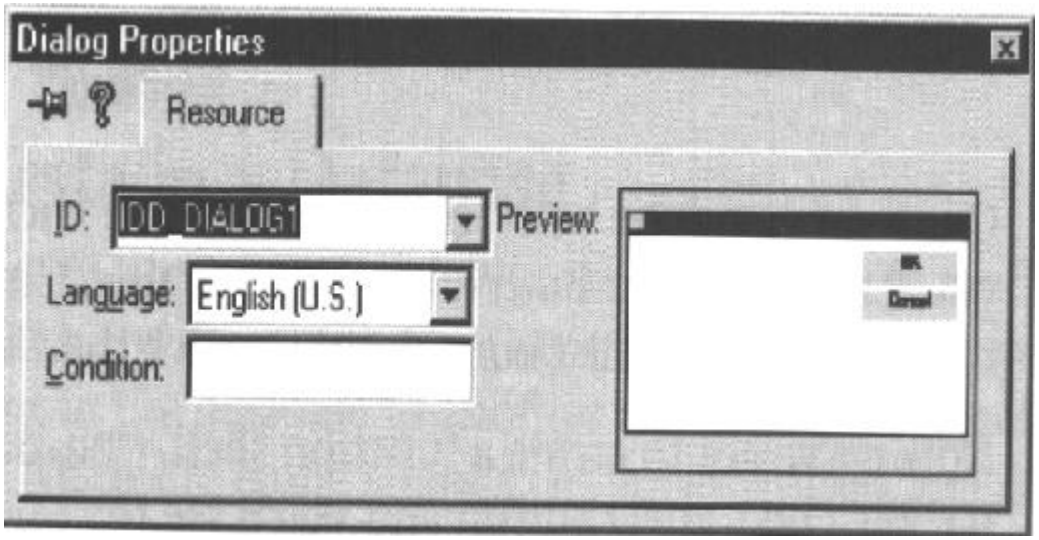


制作了按钮后，双击工具条上该按钮，就会看到一个 Toolbar Button Properties（工具条按钮属性）对话框。从 ID 下拉列表框中选择 ID_RECORD_FIND，然后单击 Close（关闭）按钮。

现在，我们已经创建了一个函数框架。按住 CTRL 并双击 ID_RECORD_FIND 按钮，会看到 MFC ClassWizard。应该选择 ID_RECORD_FIND 条目。一定要在 Class Name 字段中选择 CODBC1View。加亮 Messages（消息）列表中的 COMMAND（命令）条目，然后单击 Add Function（添加函数）。单击 OK 接受 Add Member Function（添加成员函数）对话框中的缺省函数名 OnRecordFind（我们要在本节的后面向这个函数中添加代码）。此时，MFC ClassWizard 对话框看上去应该如下图所示。



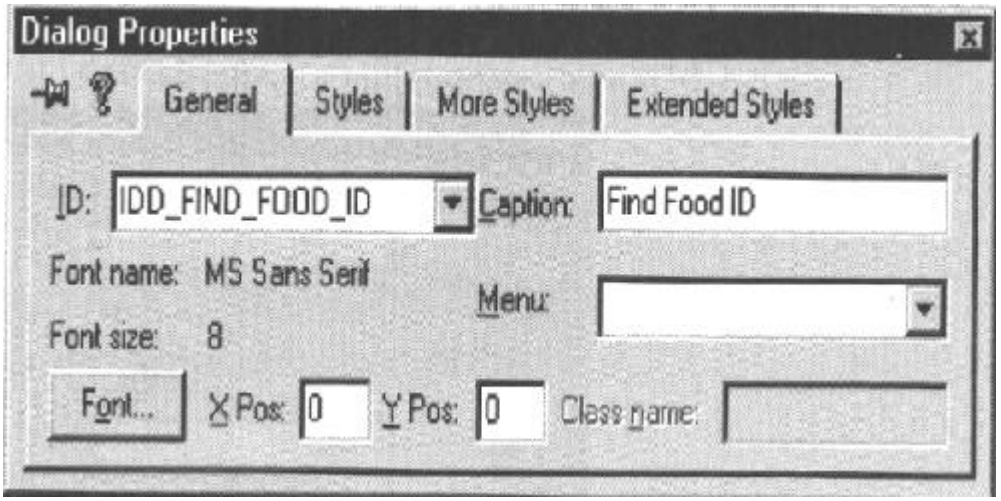
还遗漏了一件事。你打算如何询问用户要查找什么呢？完成这个任务需要一个比较小的对话框。单击OK关闭MFC ClassWizard对话框。回到ResourceView（资源视图）。右击Dialog（对话框）文件夹，从上下文菜单中选择Insert Dialog（插入对话框）。Visual C++会生成一个新的对话框。在我们继续操作之前，可以更改该对话框的一些属性。原因只有一个，为对话框赋予更容易辩认的名字。右击ResourceView（资源视图）中的IDD_DIALOG1条目，从上下文菜单中选择Properties（属性），会看到如下图所示的Dialog Properties（对话框属性）对话框。



在ID字段中输入IDD_FIND_FOOD_ID。单击Close（关闭）框完成对话框属性的更改。

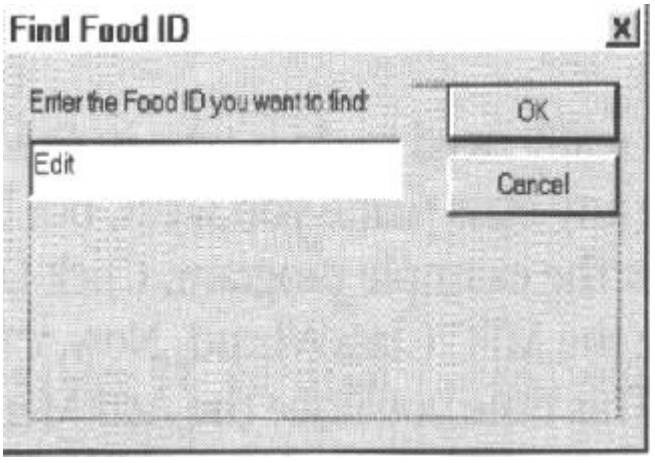
对对话框本身我们只需要做两件事。要做的第一件事是，给对话框说明一个意义明确的标题。右击对话框本身，从上下文菜单中选择Properties（属性）。这一次会看到如下图所示的Dialog Properties（对话框属性）对话框（尽管这个

对话框的名称和前面看到的相同，但是它的目的不同）。



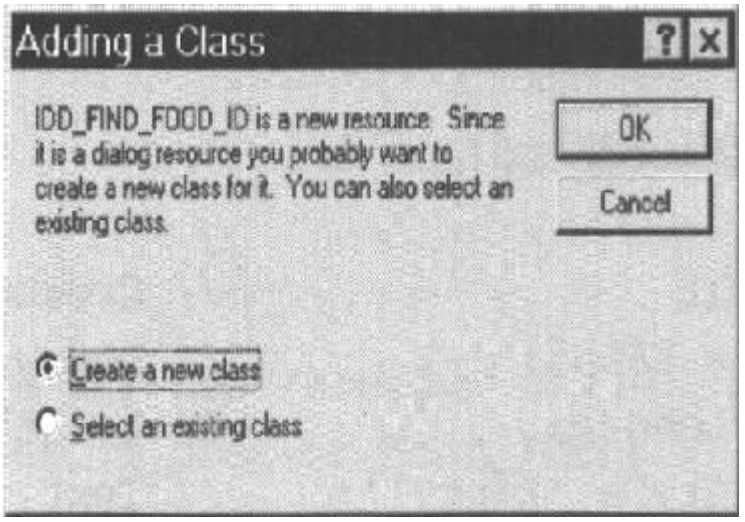
在 Caption（标题）字段中输入 Find Food ID（查找食品 ID），然后单击 Close（关闭）框完成此次更改。

现在我们要添加单行 Static Text（静态文本）控件和相关的 Edit Box（编辑框）控件来完成这个对话框。下图是最后的对话框外观。

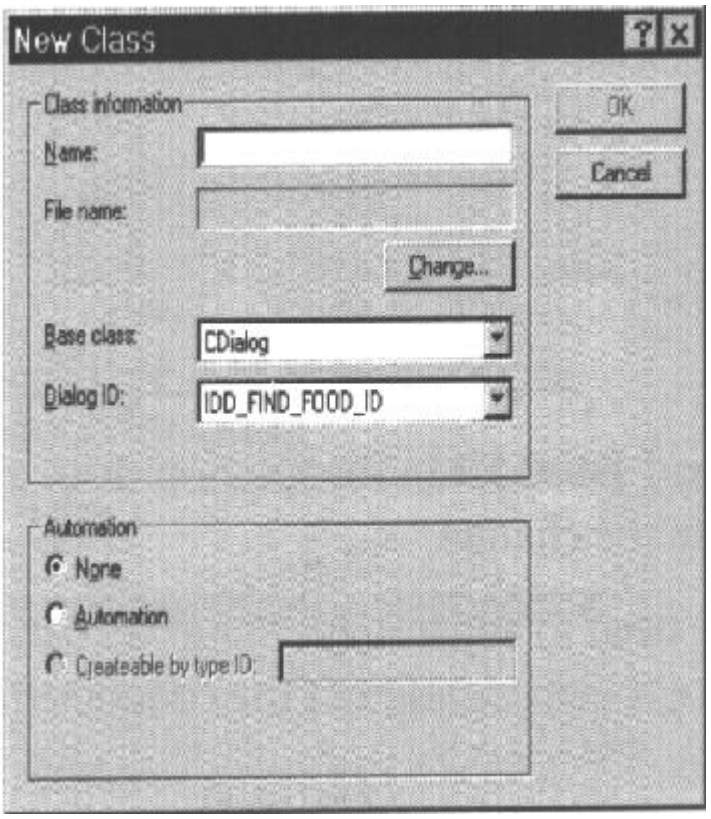


对这个对话框所做的最后一件工作是，设置 Edit Box（编辑框）控件。右击 Edit Box（编辑框）控件，从上下文菜单中选择 Properties（属性）。把 Edit Properties

（编辑属性）对话框中的 ID 属性改成 IDC_FIND_FOOD_ID。单击 Close（关闭）框完成这个动作。现在按住 CTRL 并双击 Edit Box（编辑框）控件。你首先看到的是如下图所示的 Adding a Class（添加类）对话框，它使人感到有些迷惑。



出现 Adding a Class（添加类）对话框的原因是，程序中的每一个对话框都属于某一个类。在很少情况下，你会想把对话框赋给一个现有类。就像应用程序中的 About（关于）框，每一个对话框都应该有其自己的类。单击 OK 为 Find Food ID（查找食品 ID）对话框创建新类。Visual C++显示如下图所示的 New Class（新类）对话框。



在这个对话框中需要完成的工作只是在 Name（名字）域中输入类名。可以使用任何想要的类名，但是我已经为示例程序中的类名选择了 FindFoodID。单击 OK 完成新类的创建。单击 OK 选择 MFC ClassWizard。现在，再次按住 CTRL 并双击 Edit Box（编辑框）控件，这次会看到一开始就希望看到的 Add Member Variable（添加成员变量）对话框。在 Member Variable Name（成员变量名）域中输入 m_FindFoodID。确保 Category（分类）域包含 Value（值），Variable Type（变量类型）域包含 CString。单击 OK 添加成员变量。

还记得我们以前在 CODBC1View 类中创建的 OnRecordFind()函数框架吗？现在要添加一些代码了。程序列表 6.5 显示了用来查找数据库中记录的代码。

程序列表 6.5

```
void CODBC1View::OnRecordFind()
{
 FindFoodID          oFindIt;                // Create an instance of our dialog box.
 BOOL                lFound = FALSE;         // Did we find a match record?
 int                 iStrLength;             // Length of the string.
 char*               cQuantity = "Empty1";   // Text value of Quantity field.
 char*               cStorage = "Empty2";    // Text value of Storage Life field.
 CDBVariant          varBookmark;            // Positioning bookmark.

// Display the dialog and determine which button the user pressed to exit.
if (oFindIt.DoModal() == IDOK)
  {
      // Save the current position.
      if (m_pSet->GetBookmark())
            m_pSet->CanBookmark(varBookmark);

      // Go to the beginning of the query and search
      // for the Food ID entered by the user.
        m_pSet->MoveFirst();
        while (!m_pSet->IsEOF() ^ lFound)
```

```
{
        // Check if the value is equal.
        if (m_pSet->m_Food_ID ==oFindIt.m_FindFoodID)
              lFound = TRUE;

        // Go to the next record.
        else
               m_pSet->MoveNext();
}

if (!lFound)
{
      // Display an error message if we didn't find the record.
      MessageBox("Record not found!",
            "Database Error",
            MB_OK | MB_ICONERROR);

      if     (m_pSet->CanBookmark())
              // Restore the current position if
              // the database supports bookmarks.
               m_pSet->SetBookmark(varBookmark);
```

```
    else
            // Otherwise, move to the first record.
              m_pSet->MoveFirst();
 }
 else
{
        // Display the data in the new record.
        m_oFood_ID.SetWindowText(m_pSet->m_Food_ID2);
        m_oName.SetWindowText(m_pSet->m_Nane);
        m_oPerishable.SetCheck(m_pSet->m_Perishable);

        // Process the Quantity and Storage Life fields.
          itoa(m_pSet->m_Quantity,cQuantity,10);
          itoa(m_pSet->m_Storage_Life, cStorage, 10);
          m_oQuantity.SetWindowText(cQuantity);
          m_oStorage_Life.SetWindowText(cStorage);

        // Format the price for display by adding a $--the trailing
        // zeroes were removed during the data exchange.
        iStrLength = m_pSet->m_Price.GetLength();
        m_oPrice.SetWindowText("$" + m_pSet->m_Price);
```

```
            // Get the date value and convert it to a string.
            m_oPurchase.SetWindowText(m_pSet->m_Purchase,Format("%d %B, %Y"));
        }
    }
}
```

这段程序代码中的一部分看上去与本示例程序中其它部分的代码很相似。该代码引进了一两个新的函数。应该注意的第一件事是，要检查当前的数据库位置。某些类型的 DBMS（Access 不属于这一类）支持书签。它们的作用就像是看书时用来记住阅读位置的书签一样。不过，必须做的第一件事是，用 CanBookmark()函数询问 DBMS 实际上是否支持书签。如果它支持书签，就可以用 GetBookmark()函数来保存当前的位置，用 SetBookmark()函数来恢复它。

请注意，我们的搜索例程非常简单。如果找到了用户在对话框中请求的记录，就把 Boolean（布尔）变量设成 TRUE。否则，翻到下一条记录。继续这一查找过程，直至找到与用户查找标准相匹配的第一个记录，或者是已经查询到了最后一条记录。如果你的程序采用了相似的搜索例程，那么这两种情况一定都要测试到。

注 完成一次搜索后总要复位记录指针，这样查询才能在搜索例程结束时停留在某个已知的记录位置。

此时，程序要处理两种情况。如果没有找到所要检索的记录，程序显示一个错误消息对话框。如果 DBMS 支持书签的话，程序还试图返回到前一条记录。

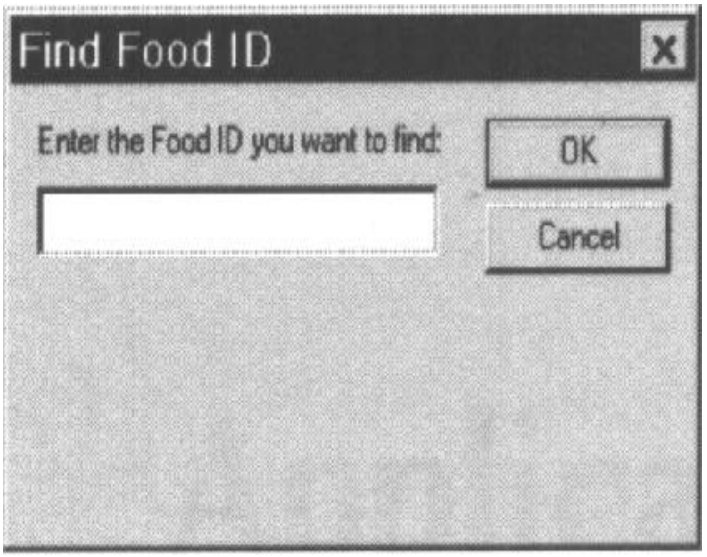
否则，程序将返回查询中的第一个记录。设置记录指针的位置是极其重要的；否则，DBMS 可能会出现异常。如果找到了记录，那么我们就处理另一种情况，在数据输入窗体中显示信息。这由本节剩余的代码来完成。你应该注意到，这些代码看上去与我们在 OnChoice2()函数末尾使用的代码十分相似。

在实际编译并运行这个应用程序之前，还要再编写一段编码。在 ODBCView1.CPP 文件的顶部，你会看到很多#include 指令。如果想使用 Find Food ID（查找食品 ID）对话框，就要在模块中包括它的头文件。添加一条如下所示的新#include 指令：

```
// Added for Find Food ID dialog support.
#include "FindFoodID.h"
```

注：在模块中用#include 指令包括你想要使用的对话框的头文件。

现在把所有的代码段都集中在一起，编译并运行程序。你会发现，在我们的小型表上，搜索例程运行得非常快，而当表变大时，它的速度就慢了下来。在大多数情况下，如果你计划处理巨大的数据库，就要购买一本大部头的包含了几千个搜索例程的书。当表包含了多达 10,000 条记录时，这个搜索例程也应该运行良好。当然，所有这些都依赖于表之间关系的复杂性以及它们所包含的字段数量。下面是搜索过程中 Find FOOD ID 对话框的外观。


Find Food ID
Enter the Food ID you want to find:
OK
Cancel

第 7 章　使用 OLE-DB 或 ADO 建立数据库应用程序

OLE-DB（对象链接与嵌入数据库）和 ADO（ActiveX 数据对象）是对 Microsoft 数据库访问技术的最新补充。这两种技术为访问所有类型的数据库提供了统一的接口，不论它出现在驱动器上的什么地方。另外，它们提供了 Microsoft 数据访问技术的最大灵活性。

阅读本章时要记住的一点是，OLE-DB 是一种基本的数据访问技术，而 ADO 是在 OLE-DB 上添加了一层便于使用的包装。另外，OLE-DB 这种产品与 ODBC 是完全脱离的。正如我们在第 6 章中所做的那样，用 DAO 创建应用程序和用 ODBC 创建应用程序有很多类似之处，用 OLE-DB 创建应用程序则要求完全不同的技术。事实上，可以把 DAO 看作是 ODBC 的高级（功能更强大）形式，只不过灵活性稍差一些（可访问的数据库引擎要少一些）。另一方面，本章会说明，OLE-DB 代表了一种全新的数据库访问方式。

在开始本章之前，假定你已经花时间阅读了第 4 章，了解了 OLE-DB 和 ODBC 之间的差别。你还要知道 ADO 是干什么的。如果你还没有阅读第 4 章，最好请补读。你还要创建我在第 5 章中已经说明如何创建的示例数据库，我们要在本章中用到该数据库。

Web 链接 在研究 Visual C++和各种支持 DAO, ADO 以及 OLE-DB 的数据库访问技术时，你一定想知道到哪里可以得到其他人的帮助。例如，microsoft.public.vc.database 新闻组中有很多关于这些技术的讨论。你还会在 microsoft.public.officedev，microsoft.public.vc.mfc 和 microsoft.public.vc.mfcdatabase 新闻组中发现一些与数据库技术相关的讨论。还有一些和非 Microsoft 的数据库技术相关的新闻组：comp.os.ms-windows.programmer.win32，comp.os.ms-windows.programmer.tools.mfc 和 comp.databases.ms-access。有趣的是，尽管在 Internet 上关于 ADO, DAO 和 OLE-DB 的讨论好像占据了很多的新闻组讨论，但是很少有专门谈论这些技术的新闻组，特别是关于 ODBC 的新闻组就更少了。显然，这就是说，为了在更为一般的新闻组（像我刚刚提到的新闻组）中寻找特定的信息，你还要花费一些额外的时间。

我们准备在本章中花很多的时间来讨论 OLE-DB 和 ADO。事实上，本章开始的方式和第 6 章很相似——建立一个简单的应用程序。但是，在这里我们会创建一个网格视图，而不是前一章中所用的窗体视图。知道如何创建多种形式的简单应用程序相当重要。否则的话，也许你就不能给客户提供正常管理其数据时所需的全部视图。

注释 因为 ADO 和 OLE-DB 在很大程度上代替了 DAO，所以本章不研究 DAO，不过，你还是可以从 Osborne Web 站点下载本章前两个示例的 DAO 版本。只要寻找 Web 站点上的 DAO1 和 DAO2 示例就可以了。我已经检查了代码，确保它可以在 Visual C++ 6.0 版上运行。你只要看一看代码本身，就会知道如何实现网格数据库实用程序的 DAO 版了。代码中的注释将有助于你对 MFC AppWizard 生成的应用程序进行必要的修改。为了使这些示例程序能够运行，必须在机器上安装 DAO。这些示例使用了 DAO 3.5 版。有几种方式可以用来检查 DAO 版本。首先，查看 Program Files\Common Files\Microsoft Shared\DAO 文件夹。如果看到了 DAO350.DLL 文件，就表明已经安装了 3.5 版 DAO。还可以查看 Control Panel（控制面板）中的 Add/Remove Programs（添加/删除程序）小程序。如果安装了 DAO 3.5 版，那么 Install/Uninstall（安装/卸载）页应该有一个 Data Access Objects(DAO) 3.5 列表项。

和上一章一样，我们要建立示例应用程序并增强其功能。在这里，我们要向示例应用程序添加报告能力。我准备说明一些 OLE-DB 专用的方式。你会发现，上一章必须使用的大量代码，比如用来搜索数据库的代码，至少有一部分现在被 OLE-DB 或 ADO 类支持的简单函数调用所代替了。

技巧 与 ODBC 不同，使用 OLE-DB 或 ADO 时不需要使用 DSN。但是，这两种技术都必须知道本地驱动器或网络上数据的特定位置。你可以用 UNC（统一命名协议）路径来减少用户找不到数据库的机会，但这种机会依然是存在的。一定要在每台机器上检查你的进行数据访问的 OLE-DB 或 ADO 应用程序。不要简单地因为你可以用 Explorer 看到相关的数据库，就假定用户可以访问这些数据了。

7.1 创建简单的网格视图应用程序

我们在第 5 章看到了两种不同的应用程序。第 6 章说明了如何创建其中的第一种，即窗体视图应用程序。在本节中，我们来看看其中的第二种，即网格视图应用程序。与第 6 章中一次只浏览查询中的一条记录不同，这回我们要一次看到所有的记录（至少要像屏幕所显示的那样多）。当然，这就要使用网格而不是用编辑框来存储各种记录值了。

你多半会发现，在谈到数据库时，网格给程序员带来了一些特殊的挑战。例如，如果你现在不能马上阅读整个查询的话，就必须跟踪查询已显示了多少记录（有一些查询可能非常巨大，内存不允许创建可以保存所有需要向用户显示的值的网格控件）。你可能还会发现，要管理所有的字段，尤其是像备忘录这样的长文本字段，还需要再做一些工作。冗长或复杂的字段可能要求一些条件或要求使用第二个对话框（与备忘录字段共同出现）。为了使这个示例更简单一些，我们要走在这些问题的前面。但是，如果数据库表包含了带这些类型

值的字段，那么记着专门考虑这些字段的处理方式是非常重要的（和第 6 章一样，我们要研究一些特殊的和日期及货币有关的格式化问题）。

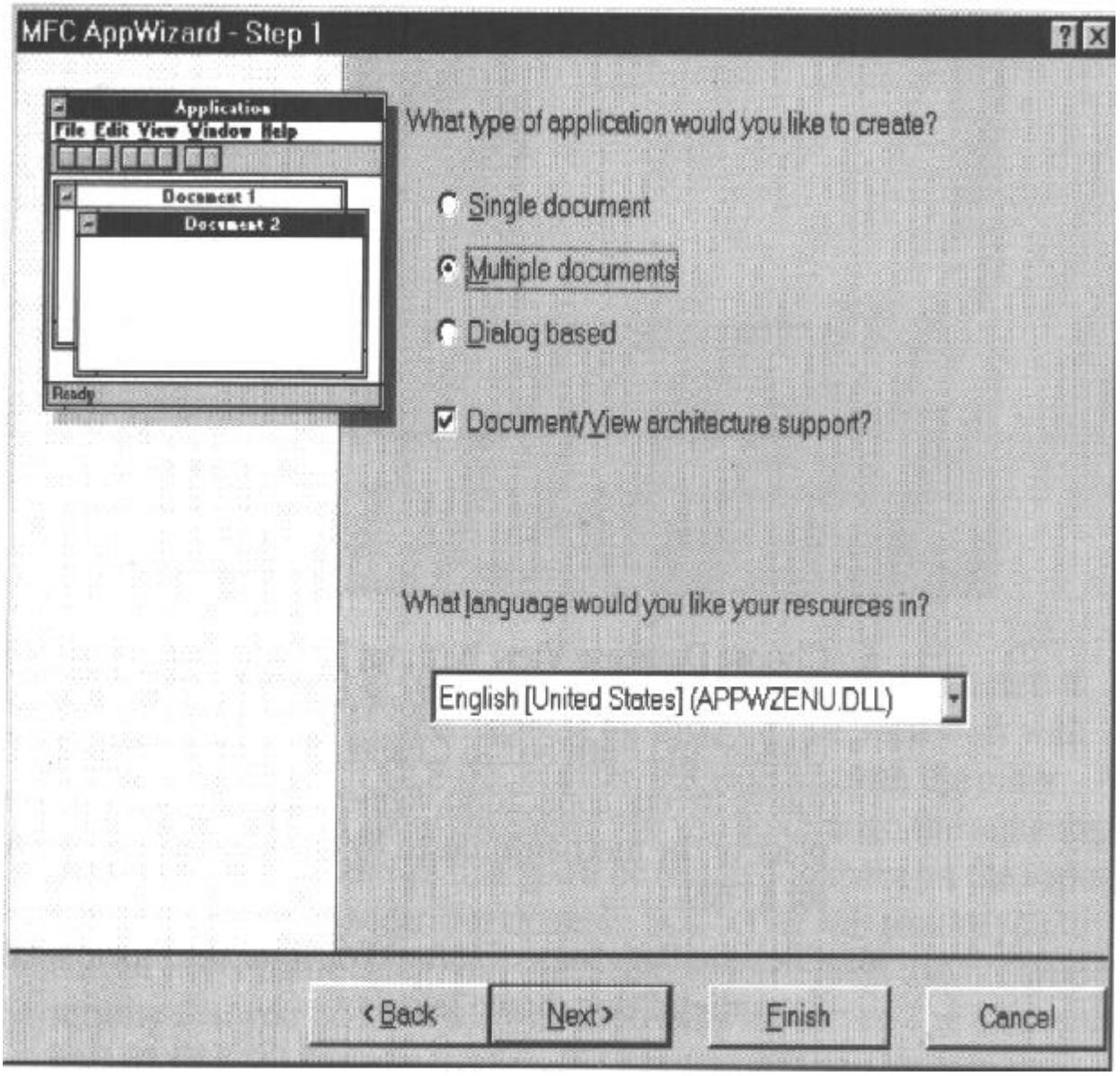
Web 链接 你可能会发现，一开始阅读本章就想进行一些额外的训练。Internet 上有很多地方提供关于用 Visual C++处理 ADO, OLE-DB, DAO 和 ODBC 的课程。例如，你会发现，http://www.unisoftinc.com/courses/ 处的 Universal Software Solutions（通用软件解决方案）提供了关于使用 Visual C++、关系数据库和更新的 Microsoft 对象技术如 DCOM 和 COM+的高级课程。另外还有 http://www.develop.com/处的 DevelopMentor（开发顾问）。它大体提供了各种有关对象技术和 Visual C++的课程（包括各种形式的数据库访问）。该公司过去常常提供一些 DAO 专用的类，但后来停止了这类业务。

创建 ADO1 工程

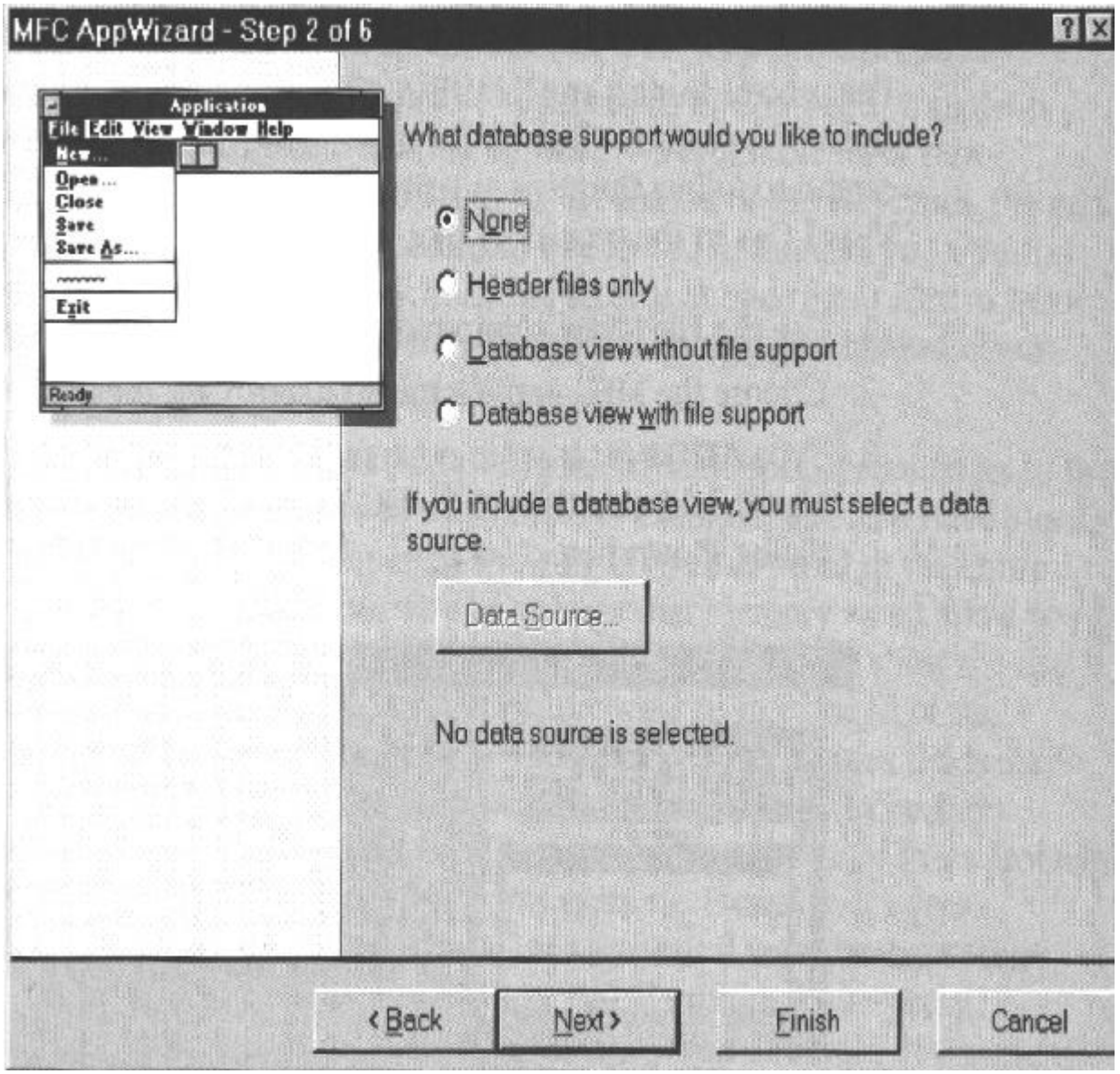
启动 OLE-DB/ADO 网格视图应用程序的过程，有点像在第 6 章中用到的过程，但是有一些重要的差别，所以要仔细遵循下述的步骤。在下面的过程中假设你已经启动了 Visual C++。

1. 使用 File|New（文件|视图）命令显示 New（新建）对话框。
2. 选择 Projects（工程）页上的 MFC AppWizard(EXE)条目。
3. 在 Project Name（工程名）域中输入 ADO1（或任何其它适合需要的名

字），然后单击 OK。你会看到如下图所示的 MFC AppWizard-Step1 对话框。

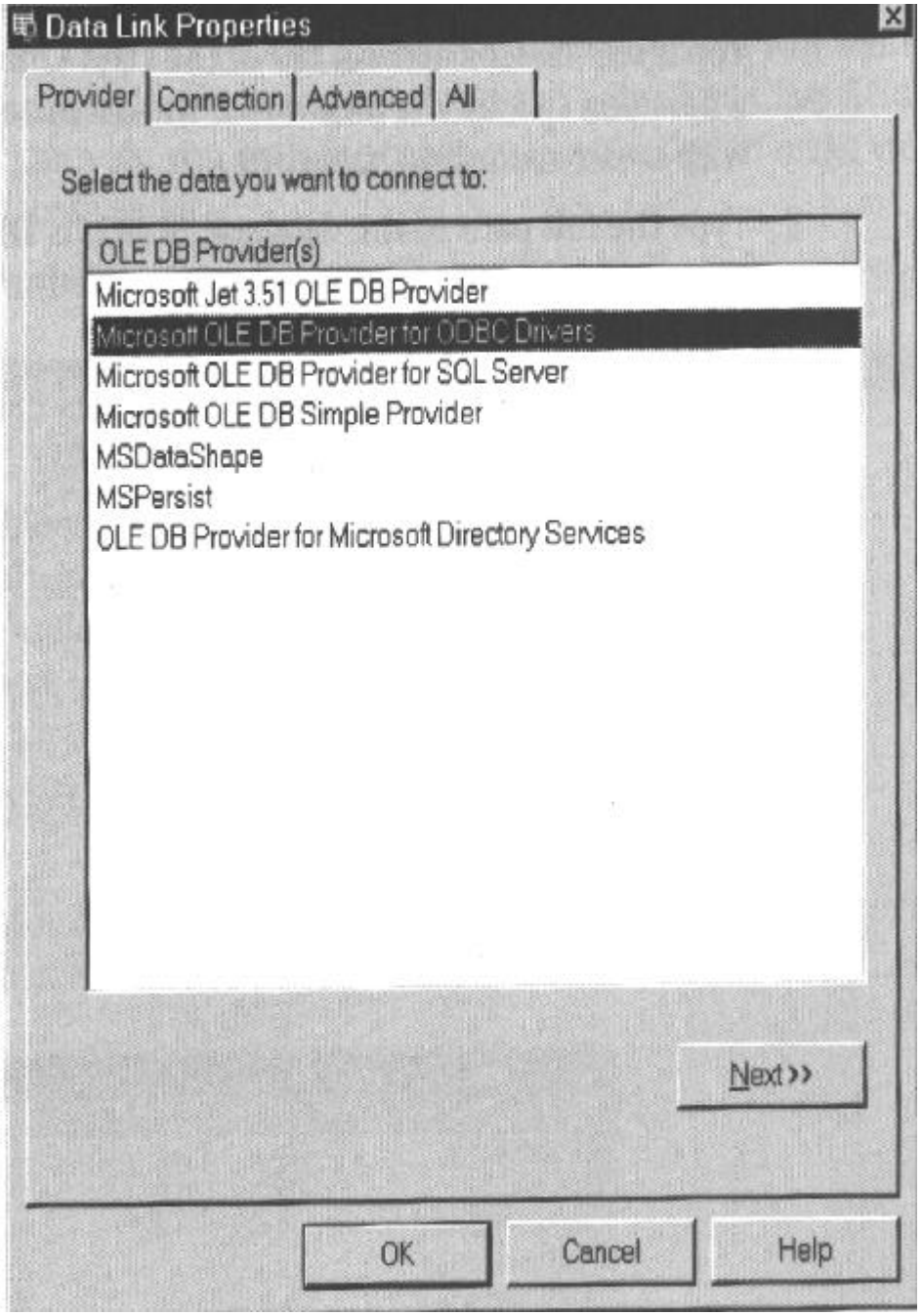


4. 选择 Single Document（单文档）选项，然后单击 Next（下一个）。你会看到如下图所示的 MFC AppWizard-Step 2 of 6 对话框。我们要做的第一件事是为应用程序选择数据库支持层次。



5. **在一系列选项中选择** Database View Without File Support **（无文件支持的数据库视图）（各种选项的概述参见第** 6 **章技巧框中的“选择数据库选项”段）。现在要为我们的应用程序选择数据源。请记住，**OLE-DB **用提供者代替了** ODBC **使用的机制；因此，我们不需要** DSN**。**

6. **单击** Data Source **(数据源) 按钮。会看到如下图所示的** Data Link Properties **(数据链接属性) 对话框的** Provider **(提供者) 选项卡。**



我们用这个对话框来选择数据源及其访问方式。在这个示例中我们使用了 Access 数据库，所以要使用 Microsoft Jet 3.51 OLE-DB Provider，不过使用其它 OLE-DB 提供者也差不多。你会经过相同的步骤，但是在不同选项卡上要求选择的项目会因提供者提供者不同而不同。例如，SQL Server 会要求你输入服务器名和其它连接条件，如访问服务器时要使用的安全类型。在获得数据库访问方面，Microsoft Jet 3.51 OLE-DB Provider 所需的信息最少。

7. 选择 Microsoft Jet 3.51 OLE-DB Provider 选项，然后单击 Next（下一个）。你会看到 Data Link Properties（数据链接属性）对话框的 Connection（连接）选项卡。在这里指定想要访问的数据库的位置以及用户名和口令。

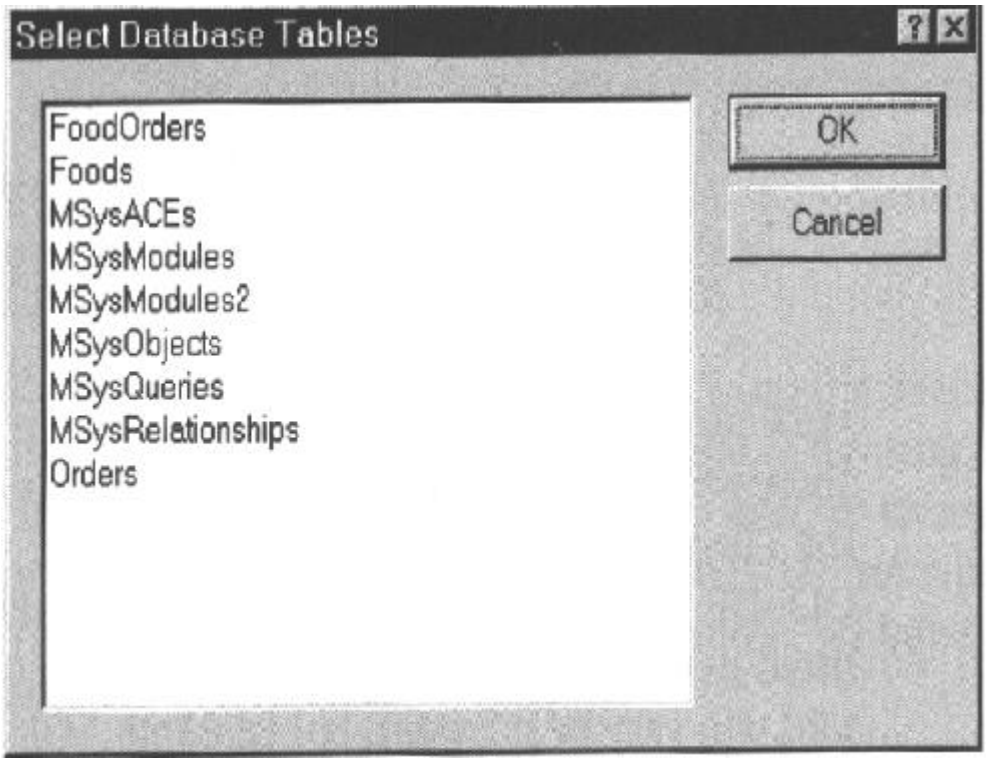
8. 输入访问数据库的完整路径，本示例中使用的路径是 DBSamp\Sample.MDB（也可以用 Browse（浏览）按钮来搜索硬盘驱动器上的数据库）。

9. 在适当域中输入用户名和口令。示例数据库对这两个值都不作要求。

10. 单击 Test Connection（测试连接）按钮。如果在 Connection（连接）选项卡中输入了正确的信息，那么你就应该看到 Test Connection Succeeded（连接文本成功）对话框。否则，要检查所有提供的值是否都正确。最常见的问题是口令输入错误（这也好办，因为 Visual C++可以用星号来代替口令）。

11. 单击 OK。现在已经连接数据库了，它并不依赖 DSN。

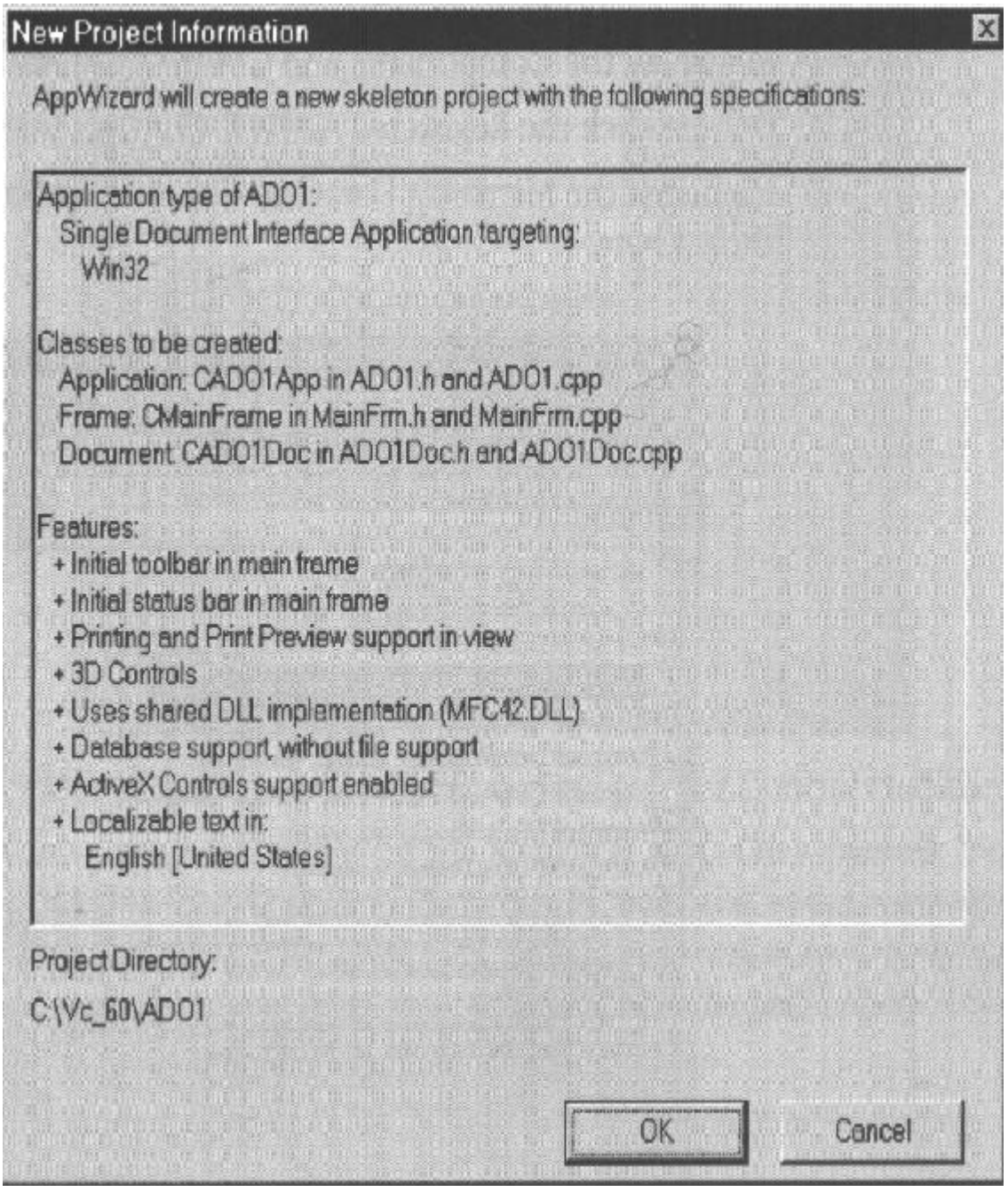
12. 单击 OK 关闭 Database Options（数据库选项）。你应该看到如下图所示的 Select Database Tables（选择数据库表）对话框。



13. **加亮** FoodOrders **选项。像第** 6 **章中的示例那样，我们也可以在需要时选择某个表。然而，通常我们使用查询来访问数据库的内容，这样才能以希望的顺序提取想要的数据。单击** OK。MFC AppWizard-Step 2 of 6 **对话框现在应该显示已经选择了数据源。**

14. **单击** Next **（下一个）四次。你会看到** MFC AppWizard-Step 6 of 6 **对话框。在这个对话框上不用改变任何设置，但是你要验证一下在** Base Class **（基类）域中是否选择了** COleDBRecordView**。这个类提供的功能让你能够更容易地浏览数据库。**

15. **单击** Finish **（完成）。你会看到如下图所示的** New Project Information **（新工程信息）对话框。**



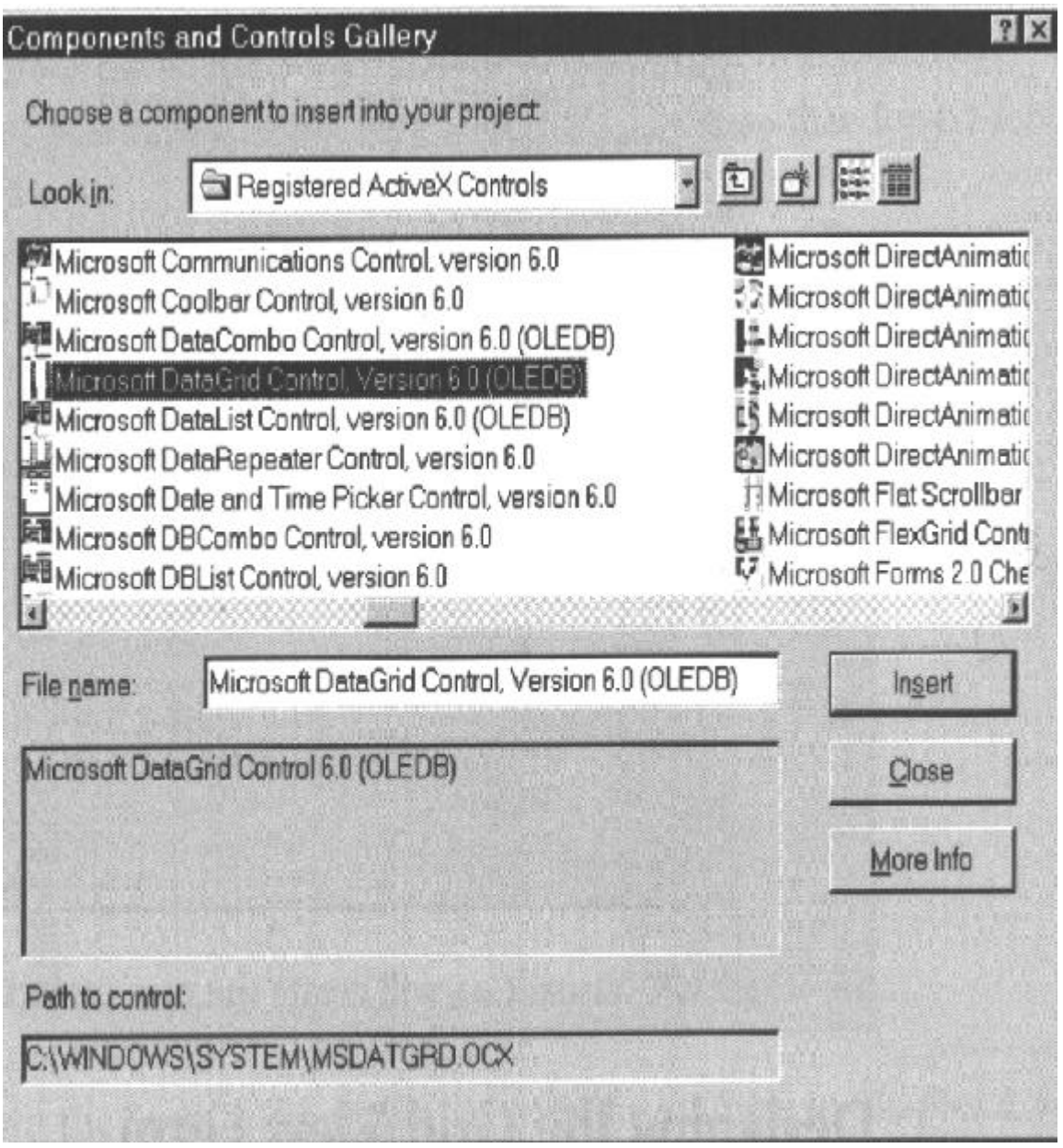
16. **单击** OK。Visual C++**创建新的工程。**

设计网格视图窗体

此时，应该看到一个名为 IDD_ADO1_FORM 的空白对话框。可以在这个窗体上放置任何东西，包括以报表形式查看所有记录的网格，或者查看每个数据字段的分离控件。现在你到了要选择如何查看信息的时候了。在本例中，我们要创建网格，但是，这里并没有任何东西要把你限制在某种视图类型上，这一点很重要。

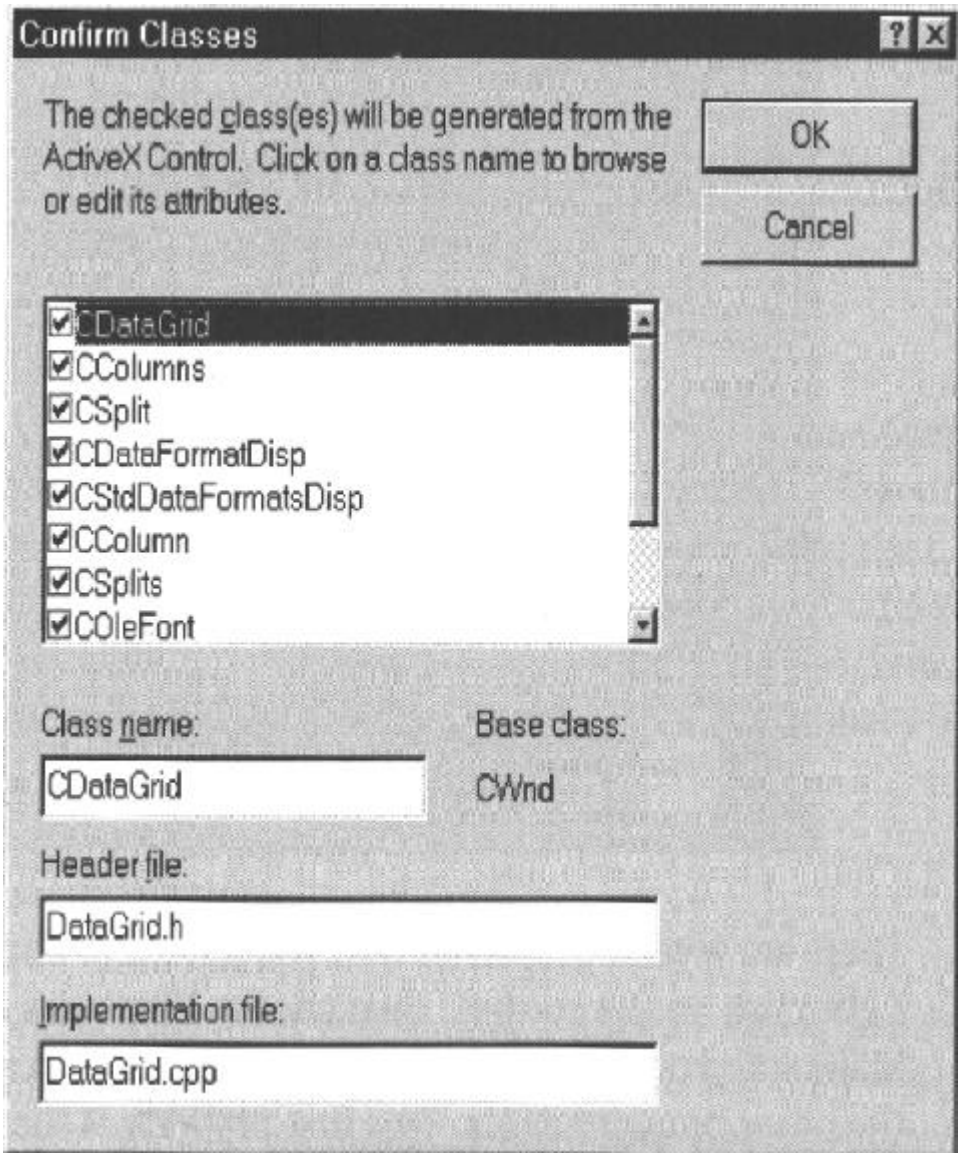
创建网格视图要做的第一件事是，向控件列表中添加一个网格。幸运的是，已经有了一个特殊的网格被设计成可以应用 OLE-DB。很明显，我们需要在窗体上做另外一些工作，但是添加了这个特殊的控件将会为我们节省许多的时间。下面的过程帮助你把该控件添加到 Controls（控件）工具条上，然后再把它添加到 IDD_ADO1_FORM 对话框中。

1. 用 Project|Add To Project|Components and Controls（工程|添加到工程|部件和控件）命令显示 Components and Controls Gallery（部件和控件库）。
2. 双击 Registered ActiveX Controls（已注册的 ActiveX 控件）文件夹。
3. 加亮如下图所示的 Microsoft DataGrid Control,Version 6.0(OLEDB)条目。



4. **单击** Insert（**插入**）。Visual C++**显示一个对话框询问是否想插入这个部件。**

5. **单击** OK。**你会看到一个如下图所示的** Confirm Classes（**确认类**）**对话框。**



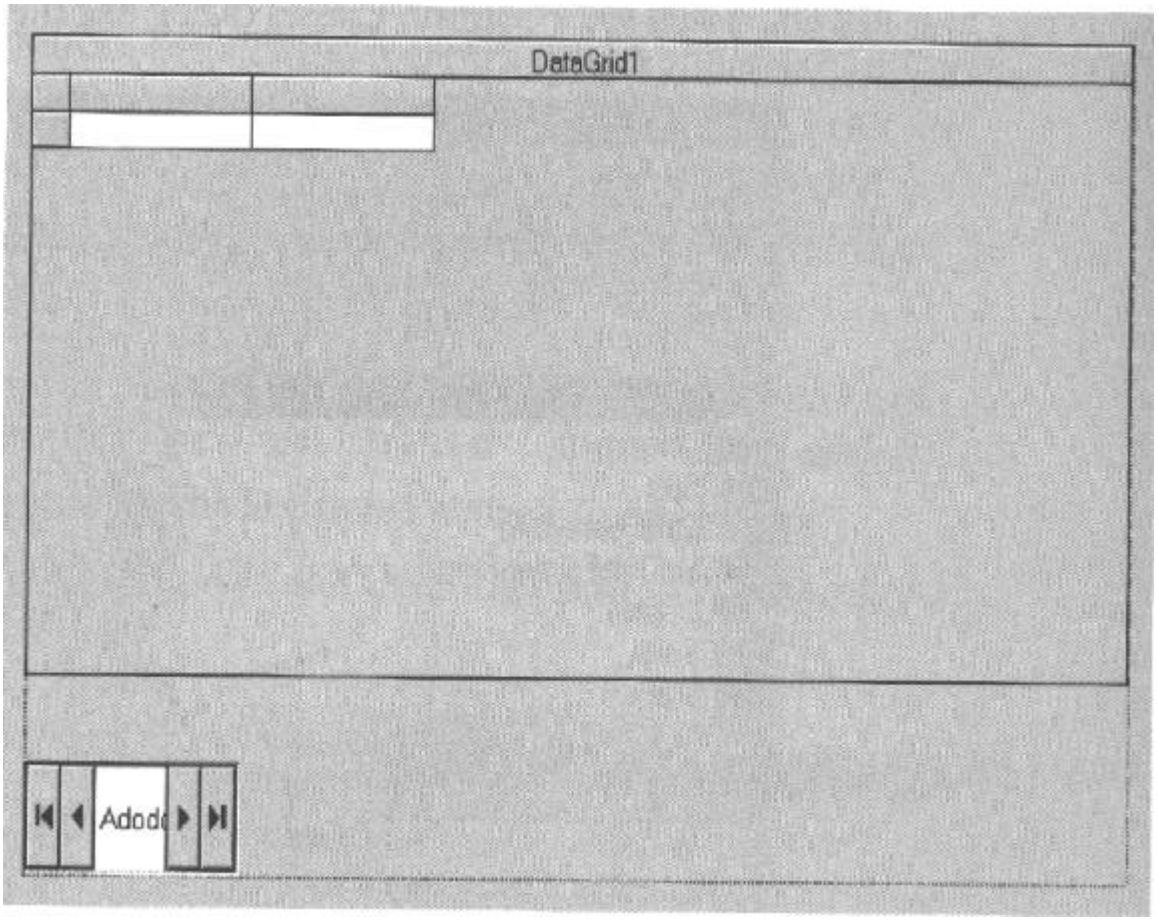
6. **单击** OK。Visual C++**把控件添加到** Controls（控件）**工具条中。**

7. **执行** Microsoft ADO Data Control,Version 6.0(OLEDB)**的步骤** 3 **到** 6。ADO Data Control **将通过我们先前创建的** OLE-DB **连接建立与示例数据库的实际连接。**

8. **单击** Close（**关闭**）**关闭** Components and Controls Gallery（**部件和控件库**）

对话框。

现在我们已经有了两个控件可以使用，把它们放在窗体上并加以配置。不要担心精确调整控件大小的问题——我们用代码来完成这项工作。你需要把这些控件放置到窗体的角上。下图是示例中使用的窗体。



在 DataGrid1 上有几个属性需要更改。只要右击控件，然后从上下文菜单中选择 Properties（属性）即可。因为有一些属性不在其它的属性页上出现，所以一定要选择 All（全部）选项卡。第一个属性最重要，因为它设置控件的数据源。把 DataSource（数据源）属性设成 IDC_ADODC1。由于我们想完整地控制数据

库以便于彻底地测试应用程序，所以你也要把 AllowAddNew 和 AllowDelete 属性设置为 True。通常情况下，Data Grid（数据网格）控件被设计成允许查看和修改记录，但不允许添加和删除记录。这也是一种对数据库保护的中等措施。此时，人们可以少许地修改数据库的内容，但不会达到破坏数据库的程度。

现在配置数据源本身。右击 ADODC1 控件，然后从上下文菜单中选择 Properties（属性）。选择 Control（控件）选项卡。注意这个选项卡上的 Use Connection String（使用连接字符串）域——如果它还未被选中，那么一定要选中它，这是我们选择数据源的地方（我们在这里选择数据源的过程与本章前面所用的过程相似）。下面是建立连接字符串的过程。

1. 单击 Build（建立）。你会看到 Data Link Properties（数据链接属性）的 Provider（提供者）选项卡。

2. 加亮 Microsoft Jet 3.51 OLE-DB Provider 选项，然后单击 Next（下一个）。你会看到 Data Link Properties（数据链接属性）的 Connection（连接）选项卡。

3. 在 Select or Enter a Database Name（选择或输入数据库名）域中输入数据库的位置。也可以用 Browse（浏览）按钮在本地或网络驱动器上定位数据库。示例程序在这个域中填入了 DB_Samp\Sample.MDB。

4. 如果需要的话，输入用户名和口令（示例数据库对这两个条目都不做要求）。

5. 单击 Test Connection（测试连接）。如果连接信息正确，你就会看到一个 Test Connection Succeeded（测试连接成功）对话框。

6. 单击 OK 两次。完成连接工作。

在应用程序设置完成之前，我们还必须再做一件事。再次打开 ADODC1 控件的 Properties（属性）对话框。这一次选择 RecordSource（记录源）选项卡。在这个选项卡的底部会看到 Command Text(SQL)字段。在这里输入从刚刚取得连接的数据库中提取数据的命令字符串。输入下面的字符串访问在第 5 章中设置的 FoodOrders（食品订购）查询所提供的所有信息。

```
Select * from FoodOrders
```

关闭 Properties（属性）对话框。如果现在编译并运行这个应用程序，就能看到网格形式表达的数据。不但如此，还可以在数据库中编辑、删除和添加记录。然而，有些地方我们需要编写一些代码，比如，实现当主窗口调整大小时 Data Grid（数据网格）控件也随之调整自身大小的功能。我们会在本章的下一节处理这些任务。

高级技巧

在应用程序中使用两个数据库连接

你可能已经注意到了，为 ADO1 应用程序使用了两个连接。我们在应用程序的设计过程中创建了第一个连接，第二个连接是在向应用程序添加 AdoDC1 之后创建的。AdoDC1（第二个连接）是查看窗体时的活动连接。如果你试着使用第一个连接（直到对菜单和工具条做了更改时才可用），不会看到Data Grid（数据网格）的内容有任何变化。

下面是C++应用程序开发者要考虑的问题。没有那么多的控件像ADODC控件那样生来就支持OLE-DB。如果计划向应用程序或报告中添加单记录视图窗体，就可能需要第二个连接来访问数据库中的数据。换言之，即使你没有打算立即使用第二个连接，也要在应用程序中包括它，至少在Microsoft 使所有的Visual C++数据绑定控件都与OLE-DB 兼容之前是这样。

使用两个或者更多的数据库连接还有另外一个原因。使用两个连接允许你在创建高速应用程序时有更大的灵活性。可以为打印窗口使用一个连接，为数据编辑窗口使用另一个连接。有两个连接意味着，在处理一个窗口时不必担心为另一个窗口所使用的连接状态。连接是分离的，所以每个窗口的连接状态都被保留。

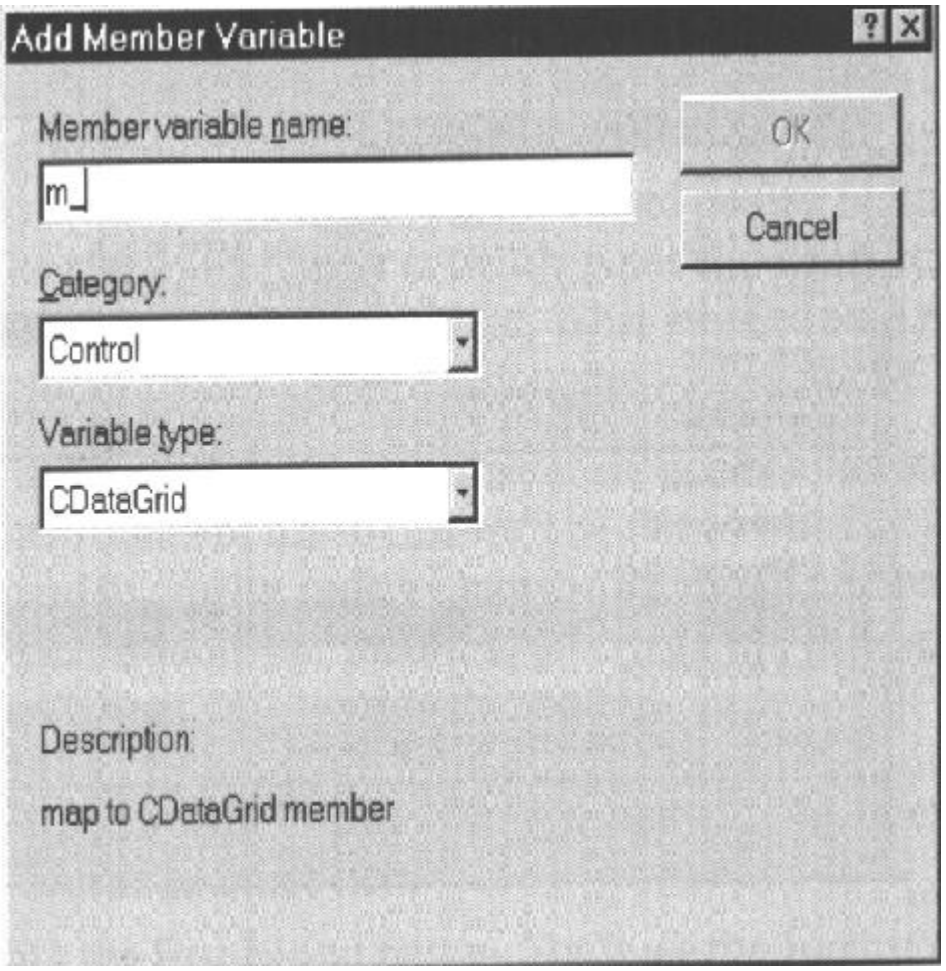
很明显，使用多个连接时存在着平衡问题。毫无疑问，使用多个连接后，应用程序的可靠性会提高一些，或许速度也有所提高。但是，获取这些特性的代价　　是使用了更多的内存。如果你在为内存有限的系统编写应用程序，那么就不必

奢侈地使用两个或多个数据库连接。内存不足的系统已经比大多数机器都慢了，用户不太可能注意到速度有所降低。然而，你需要添加另外一些代码，以确保两个窗口的连接状态信息被适当地保存，确保用户在进行一些诸如同时打印和编辑数据库操作时，不会粗心地损坏数据库。

向 ADO1 中添加一些窗体代码

示例程序在很多方面不完整，其中大部分属于可用而不完美这一类。例如，如果你现在调整应用程序的大小，窗体会变得更大，而 Data Grid（数据网格）控件不会。出现的另一个问题是，Data Grid（数据网格）控件中的所有列都被设成通用格式，这种格式可以很好地适应大多数想要显示的数据。但是，我们需要为 Boolean（布尔）值将格式设成其它的风格，而在使用通用格式时 Boolean 值显示为数字。

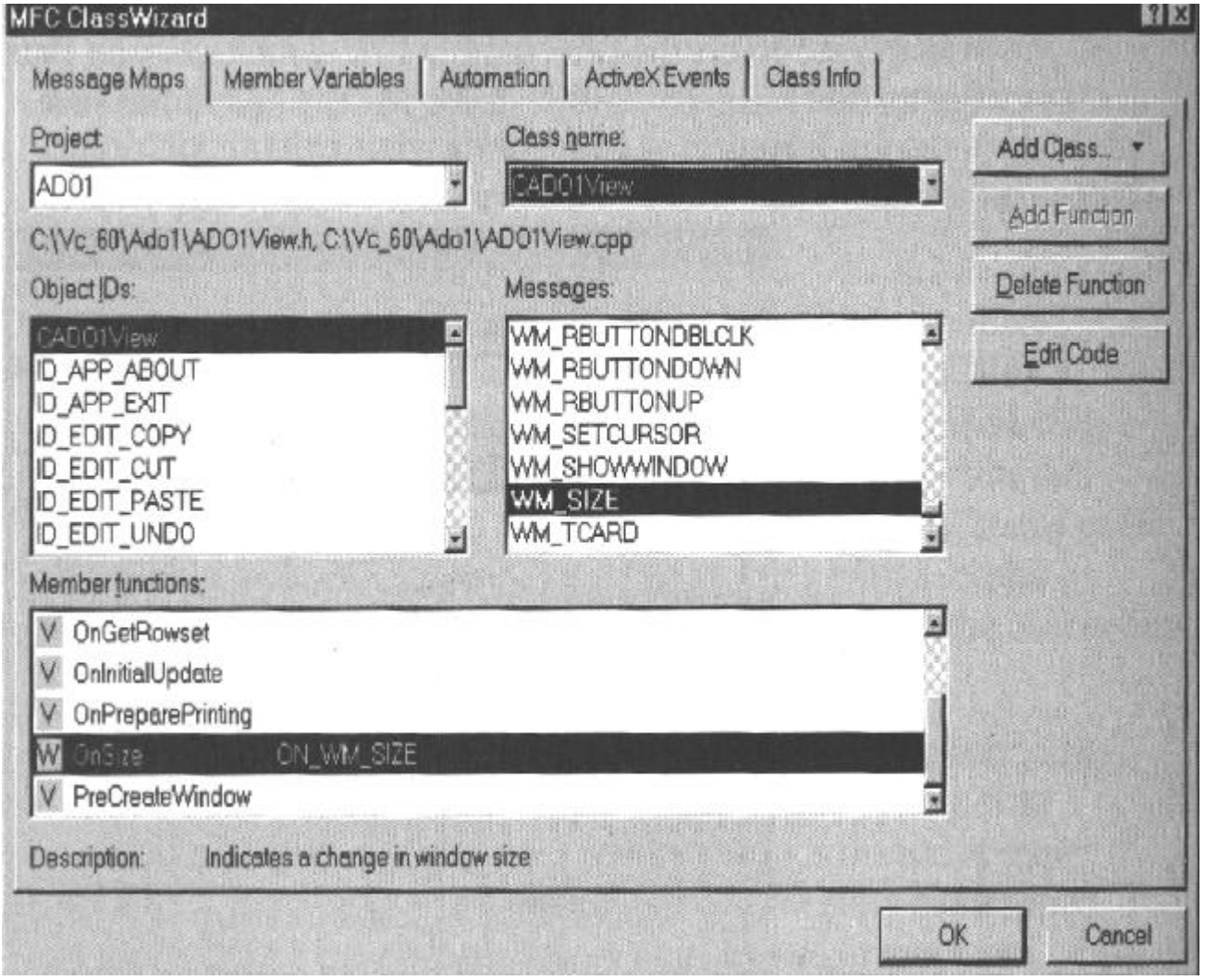
在解决这些问题之前，我们需要创建几个访问这两个控件的变量。按住 CTRL 键并双击 DataGrid1。你会看到如下图所示的 Add Member Variable（添加成员变量）对话框。



请注意，Category**（分类）域选择了** Control**（控件），**Variable Type**（变量类型）域包含了** CDataGrid**。在** Member Variable Name**（成员变量名）域中输入** dataGrid1**（完成时应该是** m_dataGrid1**）。单击** OK **完成这个过程。现在，为** ADODC1 **进行相同的操作。这一次使用** m_adoDC1 **作为变量名。**

现在添加第一个函数。使用 View|ClassWizard**（视图|类向导）命令打开** MFC ClassWizard**。第一段代码让我们在必要时改变两个控件的位置并调整其大小。在** Class Name **（类名）域中选择** CADO1View**，在** Object IDs**（对象** ID**）域中**

选择 CADO1View。在 Messages（消息）域中找到 WM_SIZE 消息。单击 Add Function（添加函数），MFC AppWizard 对话框看起来应该如下图所示。



单击 Edit Code（编辑代码），进入刚刚创建的新函数。程序列表 7.1 显示了当我们改变应用程序窗体的大小时也同时改变控件大小的代码。

程序列表 7.1

```
void CADO1View::OnSize(UINT nType, int cx, int cy)
{
    int iHeight, iWidth;                    // Control height and width.
    CRect rect;                 // Control size.

     // Perform the default action.
   COleDBRecordView::OnSize(nType, cx, cy);

   // Reposition AdoDC1.
   m_adoDC1.GetClientRect(rect);
   iHeight = rect.Height();
   iWidth = rect.Width();
   m_adoDC1.MoveWindow(cx, cy, iWidth, iHeight, TRUE);

   // Resize and reposition DataGrid1
   m_dataGrid1.MoveWindow(0, 0, cx, cy - iHeight, TRUE);
}
```

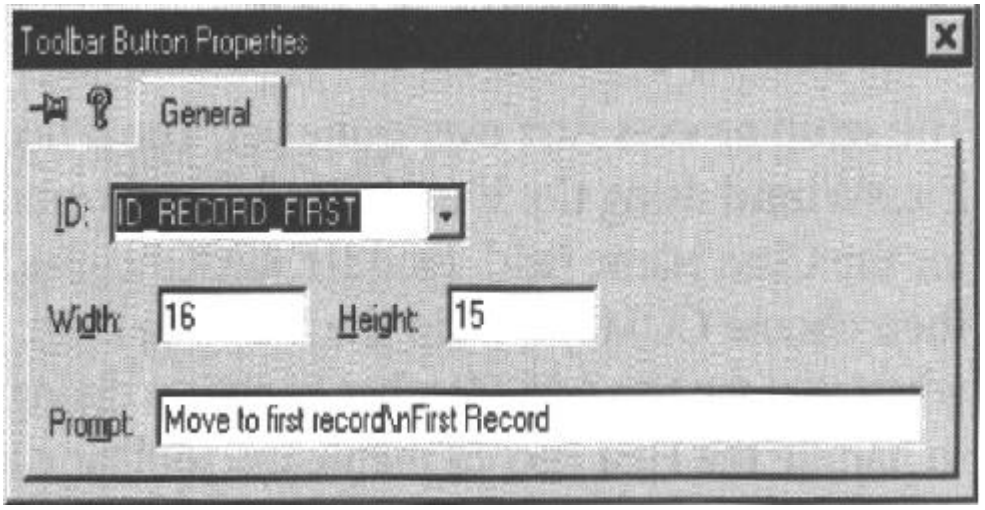
修复工具条和菜单

工具条和菜单是现在大多数应用程序的一部分，所以 MFC AppWizard 会自动地创建它们。可惜的是，自动创建过程并非总是很完美，并且在某些情况下也没有人会抱怨出现了故障。如果用 AdoDC1 控件在记录间移动，我们的应用程序会运行良好，但是如果用户想使用工具条或菜单的话，会怎么样呢？菜单和工具条好像不会做任何事情，尽管它们正在为第一个连接移动记录指针。菜单和工具条当前指向我们用 MFC AppWizard 创建的第一个连接。我们需要的是指向由 AdoDC1 提供的第二个连接的菜单和工具条。

处理菜单和工具条问题主要分两步。第一，要给它们不同的 ID，这样我们才能为它编写代码。第二，要为移动第二个连接的记录指针提供代码。下面的过程帮助你完成第一步的工作（我们从更改工具条开始）。

1. 在 ResourceView（资源视图）中双击 Toolbar（工具条）文件夹内相应项目，打开 IDR_MAINFRAME 工具条。

2. 双击 First Record（第一个记录）按钮。你会看到如下所示的 Toolbar Button Properties（工具条按钮属性）对话框。在这个对话框中无需修改很多信息，只要修改 ID 字段即可，这样我们才能添加访问第二个数据库连接所需的特殊代码。



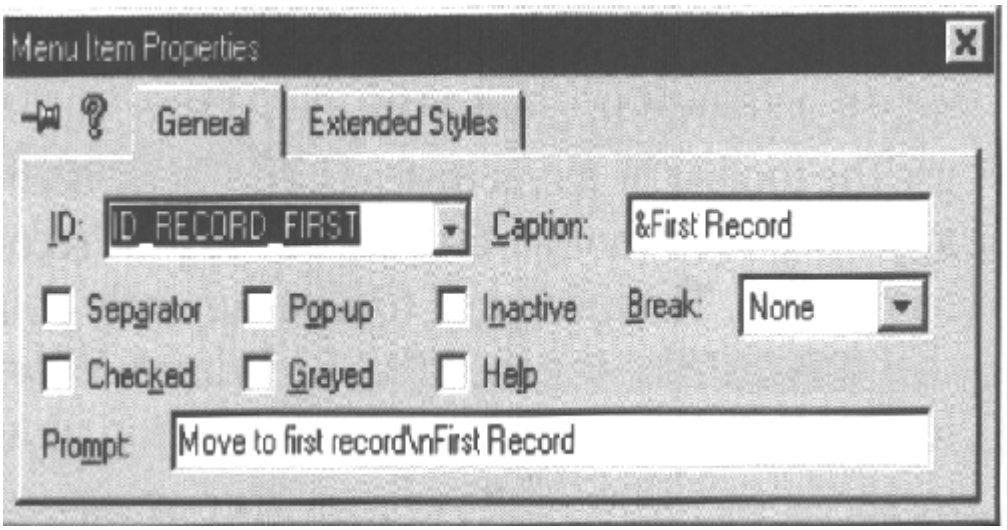
注 First Record（第一个记录）按钮的图案如下图所示：

3. **在** ID **字段中输入** ID_RECORD_FIRST2。**我们向访问第二个连接的任何对象名称中都附加上数字** 2，**使代码更易于阅读。**

4. **对** Previous Record**（前一个记录），**Next Record**（下一个记录）和** Last Record**（最后一个记录）按钮执行步骤** 2 **和** 3。

5. **在** ResourceView**（资源视图）中双击** Menu**（菜单）文件夹内的相应条目，打开** IDR_MAINFRAME **菜单。**

6. **右击** Record|First Record**（记录|第一个记录）命令，从上下文菜单中选择** Properties**（属性）。你会看到如下图所示的** Menu Item Properties**（菜单项属性）对话框。同样，除了** ID **域的值，我们不必修改其它内容。**



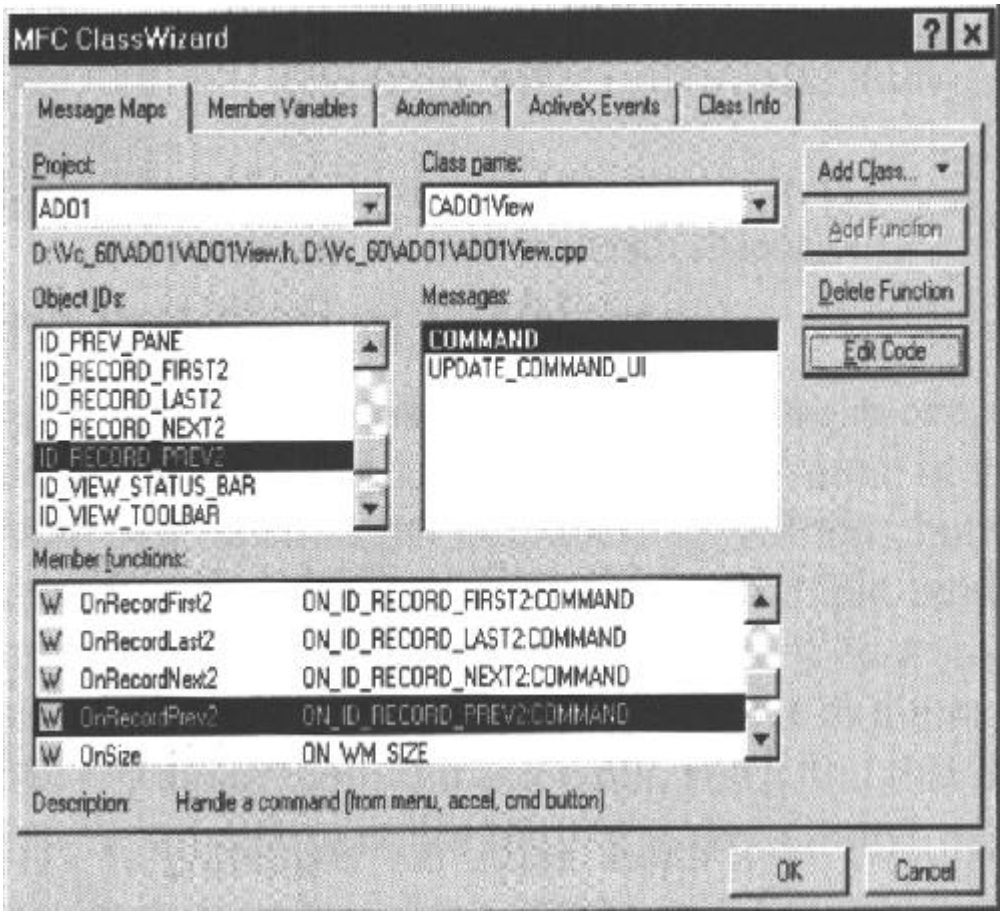
7. **在** ID **域中输入** ID_RECORD_FIRST2**。这个域段中的项目正好匹配工具条的项目，这一点绝对是必要的。否则，工具条和菜单不会互相“跟踪”。**

技巧 可以这样确保工具条和菜单名互相匹配，即从下拉列表框中选择正确的 ID 值，而不是把值输入到 ID 域中去。然而，在大多数情况下，如果是添加一个数字，就和输入值一样容易。

8. **对** Record|Previous Record**（记录|前一个记录）、**Record|Next Record**（记录|下一个记录）和** Record|Last Record**（记录|最后一个记录）命令重复步骤** 6 **到** 7**。**

设置过程还没有完成。要用 View|ClassWizard**（视图|类向导）命令打开** MFC ClassWizard**。为** Class Name**（类名）选择** CADO1View**。在** Object IDs **字段中找到** IN_RECORD_FIRST2**，然后选择** Messages**（消息）字段中的** COMMAND**（命令）。单击** Add Function**（添加函数），看到** Add Member Function**（添加**

成员函数）对话框时单击 OK。你刚才已经添加了一个处理 First Record（第一个记录）菜单和工具条项目的函数。对 ID_RECORD_LAST2，ID_RECORD_NEXT2 和 ID_RECORD_PREV2 Object Ids 重复这个过程。完成时，MFC AppWizard 对话框应该如下图所示。



此时，你已经为访问为这个程序而创建的第二个连接设置了菜单和工具条。现在要添加一些使菜单和工具条项目发挥作用的代码。在 MFC AppWizard 对话框中单击 Edit Code（编辑代码），Visual C++把你带到我们刚刚创建的新函数。程序列表 7.2 显示了需要添加的代码。请注意，#include 指令要和其它的#include

指令一起放置在代码文件的开始位置。

程序列表 7.2

```
// Include the C_Recordset class so that we
// can manipulate the recordset directly.
#include "_Recordset.h"

void CADO1View::OnRecordFirst2()
{
        // Move to the first record.
        m_adoDC1.GetRecordset().MoveFirst();
}

void CADO1View::OnRecordLast2()
{
        // Move to the last record.
       m_adoDC1.GetRecordset().MoveLast();
}

void CADO1View::OnRecordNext2()
{
        // Move to the next record.
        m_adoDC1.GetRecordset() .MoveLast();
```

```
}

void CADO1View::OnRecordPrev2()
{
        // Check to see if we can move the record pointer.
         if (!m_adoDC1.GetRecordset().GetBof())
{
         // Move to the previous record.
         m_adoDC1.GetRecordset() .MovePrevious();

         // Check again for beginning of file.
         if (m_adoDC1.GetRecordset().GetBof())
  {
               // If we are at the beginning of the file,
               // advance the record pointer to the first record.
              m_adoDC1.GetRecordset() .MoveNext();
         }
    }
}
```

程序列表 7.2 **中的大多数代码都很简单。我们用** m_adoDC1 **内存变量得到对记录集（用户当前正在观看的记录集）的访问。我们一开始访问记录集，就可以把记录指针移到适当的记录位置上。**

现在来看看 OnRecordPrev2()函数。由于我们在查阅文件起始记录时没有执行任何缺省动作，所以这个函数要求的代码比其它函数要稍微多一些（如果查阅到文件的末尾，应用程序自动向表中添加一个新记录）。所有这些额外的代码帮助你避免可视化问题，它可能会导致调用许多支持。如果记录指针在文件的开始位置遇到了文件结束，那么网格控件上的小型记录指针指示器就会消失。即使网格和 ADODC 都在正常工作，对用户来说，这可能表示发生了某种错误。要避免这个问题很简单：只要检查文件指示器的开始，并适当改变记录指针的位置。

高级技巧

避免安装问题

数据库应用程序是你可以创建的最复杂的应用程序之一。人们不但希望这些应用程序在稳定性和坚固性方面有更大的提高，而且应用程序本身的整体规模也越来越庞大了。例如，大多数的数据库应用程序要求某种网络的支持。判断出数据库应用程序在要求支持方面比任何程序都多并不难，结果你会发现，即使像我们刚刚创建的这样简单的数据库应用程序，都要求众多的支持文件。

创建一个安装程序时会碰到的第一个问题是，找出所需的全部文件。通常，在获得完整的文件列表之前，要检查多个信息源。必须寻找的两个文件集是，厂商专用的数据库访问驱动程序和操作系统支持文件。可能还需要配置实用程序和依赖于所使用的 DBMS 的其它程序文件。厂商通常会提供完整的厂商专用文件，它可能隐藏在文档的某些角落中，一定要花些时间把它们都找出来。在某些情况下，厂商还会慷慨地提供一系列通用操作系统文件，但是在假定厂商提供了正确或最新的信息之前，一定要检查一下 Microsoft 的 Web 站点，看看有没有最新的驱动程序。表 7.1 提供了一系列完整的文件，你在处理依赖 Microsoft Jet Engine 的数据库应用程序时需要它（这个表是特别为 DAO 设置的，但也可用于其它形式的数据库访问）。

表 7.1 Microsoft Jet Engine(DAO)所要求的文件

用途	3.0 版	3.5 版
Microsoft Jet Engine 库	MSJT3032.DLL	MSJET35.DLL
Microsoft Jet Database Engine 出错 DLL	MSJTER32.DLL	MSJTER35.DLL
Microsoft Jet Database Engine 国际化 DLL	MSJINT32.DLL	MSJINT35.DLL
Visual Basic for Applications 开发环境——表达式服务加载程序	VBAJET32.DLL	VBAJET32.DLL

续表

Visual Basic for Applications Runtime——表达式服务	VBAR2232.DLL	VBAR332.DLL
Visual Basic for Applications 对象库	VEN2232.OLB	N/A
Microsoft(r) Red ISAM	MSRD2X32.DLL	MSRD2X35.DLL
*Microsoft Jet xBase ISAM	MSXB3032.DLL	MSXBSE35.DLL
*Micorsoft Jet Paradox ISAM	MSPD3032.DLL	MSPDOX35.DLL
*Micorsoft Jet Text ISAM	MSTX3032.DLL	MSTEXT35.DLL
*Micorsoft Jet Excel ISAM	MSXL3032.DLL	MSEXCL35.DLL
*Microsoft Jet Lotus 1-2-3 ISAM	MSLT3032	MSLTUS35.DLL
*Microsoft Jet Exchange ISAM	N/A	MSEXCH35.DLL
Microsoft Wingman 库	MSWNG300.DLL	MSWNG300.DLL
Microsoft ODBC 桌面驱动程序包	ODBCJT32.DLL	ODBCJT32.DLL
Microsoft ODBC 桌面驱动程序包	ODBCJI32.DLL	ODBCJI32.DLL
ODBC Helper 函数 DLL	ODBCTL32.DLL	ODBCTL32.DLL
*Microsoft RDO（远程数据对象）Engine 控件	N/A	MSRDO20.DLL
*Microsoft 客户机光标 DLL	N/A	RDOCURS.DLL
*Microsoft 复制库	N/A	MSREPL35.DLL

* 根据计划要提供的数据库支持，你可能需要的可选文件（例如，如果计划访问像 FoxPro 和 dBASE 创建的 xBase 数据库，就只需要 xBase ISAM DLL）。

另一个安装问题是数据库的存放位置。某些程序员会假定用户将需要直接的网络访问，并且管理人员会把数据库放在服务器上的特定位置。不要做这样的假定，尤其是在今天这个 Internet 主宰的市场中。服务器访问可能要经历一次 Internet 连接，甚至可能来自存储于数据库中的记录的本地子集。换言之，除非你也可以保证亲自把数据放在预期的位置，否则不要做任何关于数据会存储在哪里的假设（即使这样，管理也可能决定它随时需要 Internet 的连通性）。

我们再回过头来谈一谈支持文件。你们大多认识到了，在应用程序依赖特定版本的公共支持文件（如 MFC 运行文件）的时候会有问题出现。创建数据库应用程序时，这个问题会大大增多。首先，你还要考虑 MFC 的运行。添加到这些支持文件中的是数据库访问所需要的公共操作系统文件和你所需要的厂商专用文件。现在你要及时了解三个产品版本，它们随时都可能会改变。

解决这个问题的办法是，一定要检查安装过程中用户安装在系统上的每一个文件的版本（现在你知道了为什么一些安装程序会工作上几个小时！）。如果一个文件的版本和为这个应用程序所使用的文件不匹配，那就要征得用户的同意，用所需要的文件改写这个文件。现在，有一个问题，用户都乐于假设新版本会更好一些。另一方面，如果应用程序依赖于较早版本的文件，那它有可能在新版本下运行不了。你要特别检查一下新版本的文件，并在消息中放入某种关于应用程序可能在新版本下运行不了的警告（除非你事先就知道它可以运行）。

技巧 每次应用程序启动时，很多的数据库程序员都会执行检查文件版本号的例程。在一些情况下，检查应用程序所使用的每一个文件是没有必要的。但是，检查公共或共享文件的版本号却是一个好主意。用户可以安装另一个应用程序，它改写了你的应用程序所要求的文件版本。确保应用程序可以满足需要的唯一办法是，在程序初始化过程中检查这些文件。在应用程序实际启动之前就中断，比应用程序一启动和数据库一打开程序就崩溃要好一些。

其它的安装问题和通常安装程序的设计方式有关。例如，大多数安装程序假定，如果在用户的硬盘驱动器上有足够的空间来存储应用程序的所有文件，应用程序就会正常运行。可惜的是，在数据库应用程序的情况下，你还必须考虑像记录集的本地拷贝这样的事情，因为记录集可能非常大，内存存储不了。换言之，要在安装程序中适当调大硬盘容量需求以满足应用程序的处理需要，而这一工作在安装大多数其它类型的应用程序时没有必要考虑。

7.2 向应用程序添加报告

人们都希望为数据库应用程序至少创建一个报告。即使是像我们在本章创建的这样一个小的应用程序，也可能因报告类型丰富多采而容光无限。例如，概述报告通过摘要相似的数据来从整体上显示数据库的内容（甚至摘要的方式也可以创建多个报告），细节报告则显示所有的记录。除了报告所包含的数据及其出现的顺序以外，还有一些事情要考虑，比如从审美观点来讲的报告的外

观。你可能不想花时间用最佳的文字和图形为工作组会议打印报告，但我可以肯定，经理会议通常需要这样的报告。

我无法在一个章节中告诉你创建报告的每一个要点。本章要讨论的是关于创建一批复杂报告的技巧，其中包括浏览数据的各种方法，并介绍一个如何完成上述部分目标的程序设计示例。

存在打印困难吗

创建一个你所需要的报告可能有点困难，在某些情况下也确实是如此。处理一项工程时，可能五个经理都想把他们自己的一组报告添加到应用程序中去，而你知道，在允许的时间范围内，你确实只有工夫创建一两组报告。不要失望，如果花点时间分析一下经理们在期待些什么，并且在打印过程中增加一点灵活性的话，就可能根本没有必要去编写所有的报告了。

当你试图减低应用程序的复杂性时，应该做的第一件事是，把所需的报告分解成几个简单的需求。以这样的列来创建一个表是个好主意，即按列说明简单的需求，然后在旁边列出报告。如果在报告中找到了某种对比方式，就有可能把几个报告合并成一个，然后按需要添加配置对话框。下面是关于如何把报告分解成一组部件的一些想法。

- 排序　每一个报告都要求某种排序。否则，就会形成一系列没有人能够理解的紊乱数据。请记住，数据库的一个主要目标是，把所包含的数据组织起来，使它更容易被理解。你的报告也应该是一样。排序是一种把两个或多个报告合并到单个程序设计任务中的最容易的方式。

- 分组　有时候你想要细节报告在单个标题下组织数据。例如，你可能在合同数据库中用 ZIP 代码创建分组。ZIP 代码作为标题显示，该 ZIP 代码区中的所有合同都在该标题下按顺序排列。谈到组织数据时，分组和排序并没有太大的差别。但是，即使打印输出的主要部分都相同，把两个报告和不同的组结合起来也是非常困难的。原因很简单：把各种报表头组织在一起将使得打印例程十分复杂。然而，你可以把一个包含分组，另一个不包含分组的两个相似报告结合起来。此时，你只需要在程序中提供一个开关，用它打开标题或者关闭标题就可以了。
- 分组合计/摘要　许多报告会提供这样的脚注，整个报告或一组有关的数字在那里被记录下来。脚注还有很多其它的统计用法，但数字记录是最普通的类型。在有组标题的情况下，如果有可能，要把两个有不同合计或者摘要的报告结合起来是很困难的。但是，可以把未使用任何类型脚注的报告和使用了脚注的报告结合起来，只要它们的列是相同的即可。
- 报告外观　如果你有两个报告使用相似的数据但具备不同的外观风格，那么可以试着把它们结合起来。如果唯一的差别只是文字所用的字体或者是像添加到报告顶端的公司徽标这样的小窗口，那么创建两个不同的打印例程就没有什么实际意义了。
- 细节数据的层次　某些报告只是其它报告的简单综述。换言之，有一些报告完全只使用父数据。通过以某种方式摘要子记录来提供综述信息。把综述报告和显示数据库详细内容的报告结合起来相当容易。当然，这

也存在着报告中包含什么内容的问题。例如，如果一个报告以售货员顺序显示本月的每一张发票，而另一个报告显示各位售货员的总销售额，那么你就可以把这两个报告合成为一个。使用一个简单的开关来判断用户是要摘要数据或者要按原样打印子表中的每一个记录。

- 筛选　在数据库中寻找部分而非全部的记录，对人们来说十分普通。如果两个报告唯一的差别是用来决定该筛选哪些记录，那么你就可以把这些报告结合起来。筛选是你对 DBMS 所做的查询的函数，而不是你编写的代码的函数。换言之，你要做的只是更改一两行代码。

此时，你可能会感到奇怪，为什么不创建一个带有众多配置对话框的单个报告，让用户来产生他们所需的报告呢？不这样做有几个充足的理由。第一，大多数用户会对创建报告所需的大量对话框不知所措。你会发现，和自己设计报告相比，你还要花大量的时间让用户学会提高创建报告的速度。第二，你要引进自定义报告，还要为用户创建某种把那些报告存盘的方法，这样他们以后才能在无需很多支持的情况下，创建相同的报告。

避免通用报告的另一个原因是与安全有关的风险。如果提供给用户太多不同的创建自定义报告的方法，你可能会发现，这也带来了很多的安全漏洞。管理上通常要保证公司的秘密数据不会误入他人之手。这意味着，除了要增强数据库本身的安全性外，还要增强已打印文件的安全性并把它牢牢地控制住。

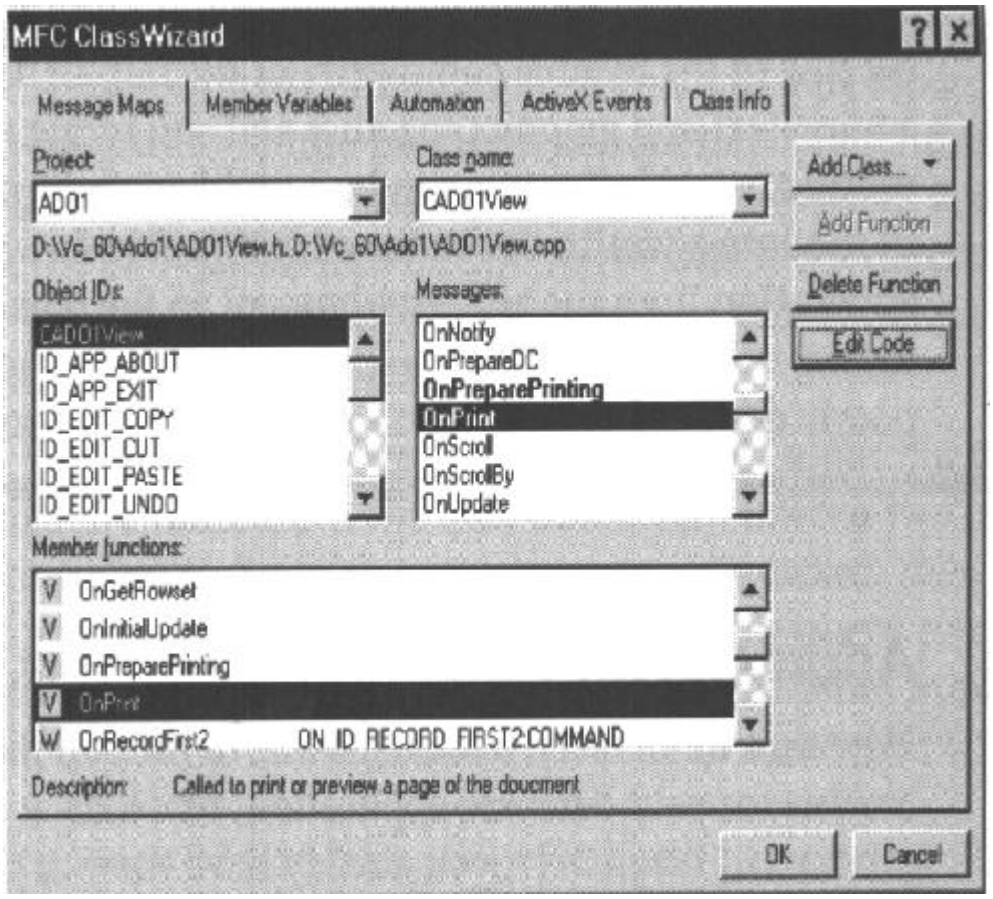
可移植性 当使用 Windows 碰到打印机不兼容的问题时，不要误以为是安全性问题。当然，Windows 为打印例程提供了一致的程序设计 API，这使编程工作变得更加容易。你还得到了打印驱动程序支持的明显优点——至少不必像 DOS 环境下的程序员那样，还要编写自己的驱动程序。但是，优点也就这么多。厂商仍然有可能提供单薄的驱动程序，这使得很难从看上去正确的应用程序中得到正确的输出。你还会发现，由于产生输出的相对复杂性，与其它开发人员相比，这些问题会更多地发生在数据库程序员身上。仅仅把文本和图形添加在一起并保持各种表格元素对齐就可能存在某些问题。可以这么说，把应用程序从测试环境转移到实用环境中时，会遇到大量与打印机有关的问题。即使是世界上最好的测试也不能帮助你避免设计粗糙的打印驱动程序所带来的问题以及其它各种各样的兼容性问题。

给应用程序添加打印功能

看过了为数据库应用程序编写打印例程的复杂性，现在我们来看一看解决这些问题的办法。幸运的是，在我们使用的 ADO1 应用程序中，现在已经有了所需的三个函数框架。我们只看一看 CADO1View 类。有三个函数：OnPreparePrinting()，OnBeginPrinting()和 OnEndPrinting()。第一个例程让你设置打印机并完成其它的预备任务，如在查询中保存当前的记录数，这样才能在打印作业完成后返回到那里。第二个例程 OnBeginPrinting()让你准备设备的上

下文信息。可以把设备上下文看作是画家的调色板——用它来绘制想要发送给打印机的信息。在本示例中，我们会经常碰到 Windows GDI（图形设备接口）这么一个特别的东西，所以你迟早会很好地理解设备上下文究竟意味着什么。最后，OnEndPrinting()例程让你把应用程序的总体设置恢复成打印前的条件。例如，你应该在这里把查询指针返回到其打印前的记录处。

哎，还遗漏了一个函数。我们还没有用来向打印机发送数据的打印例程。我们要做的第一件事是使用 MFC ClassWizard 向程序中添加一个 OnPrint()函数。OnPrint()函数包含实际把打印信息输出到打印机所需的代码。其它与打印有关的任务也调用这个例程，如大多数应用程序提供的 Print|Preview（打印|预览）命令。使用 View|ClassWizard（视图|类向导）命令打开 MFC ClassWizard 对话框。在 Class Name（类名）域中选择 CADO1View，加亮 Object IDs（对象 ID）列表中的 CADO1View 和 Messages（消息）列表中的 OnPrint。单击 Add Function（添加函数）。Member Functions（成员函数）列表中应该添加了一个新函数，如下图所示。



单击 Edit Code **（编辑代码）按钮。现在向应用程序中添加一些与打印有关的代码。程序列表** 7.3 **包含了需要向** OnPrint()**函数添加的代码。**

程序列表 7.3

```
void CADO1View::OnPrint(CDC*DC, CPrintInfo* pInfo)
{
    int          iRowCount = 1;              // Current print row count.
    LPSTR        lpstrRow ="EMPTY1";         // Text form of row count.
```

```
    CString     cLine;              // One line of printed output.
    CPen        oPen;               // Pen for drawing.
    CBrush      oBrush;             // Brush for shading.
    CFont       oTextFont;          // Font used for displaying text.
    CFont       oHeadFont;          // Font used to display the
heading.
    CFont       oColFont;           // Font used to display column
headings.
    LOGFONT     lfFont;         // Font characteristic structure.
    CSize       oFontSize;          // Size of a font.
    COLORREF    clrRef;         // Colorstructure.
    int         iRowPos = 120;      // Row position on printed page.
    int         iTextHeight  =0;    // Current text height.
    CRect       oDrawRect;          // Drawing area for printer.
    int         iRecNumPos;         // Record number position.
    int         iFoodIDPos;         // Food ID position.
    int         iNamePos;           // Name position.
    int         iPerishablePos;     // Perishable position.
    int         iPricePos;          // Price position.
    int         iPurchasePos;       // Purchase Date position.
    int         iQuantityPos;   // Quantity position.
```

```
int             iStoragePos;            // Storage Life position.

// Get the drawing area for our print routine.
ODrawRect = pInfo->m_rectDraw;

// Create a pen and select it into our device context.
clrRef = 0x00000000;
oPen.CreatePen(PS_SOLID, 2, clrRef);
pDC->SelectObject(&oPen);

// Create a brush and select it into our device context.
clrRef = 0x00C0C0C0;
oBrush.CreateSolidBrush(clrRef);
pDC->SelectObject(&oBrush);

// Create a heading font and select it into our device context.
oHeadFont.CreatePointFont(240, "Arial", pDC);
pDC->SelectObject(&oHeadFont);

// Display our heading.
oFontSize = pDC->GetOutputTextExtent("The ABC Company");
```

```
pDC->Ellipse(500,
        iRowPos - (oFontSize.cy / 2) - 10,
        oDrawRect.Width() - 500,
        iRowPos + (oFontSize.cy / 2) + 10);
pDC->SetBkMode(TRANSPARENT);
pDC->TextOut((oDrawRect.Width() - oFontSize.cx) / 2,
        iRowPos - (oFontSize.cy / 2) - 10,
        "The ABC Company");
pDC->SetBkMode(OPAQUE);

// Creat the appropriate space.
oHeadFont. GetLogFont (& lfFont);
iRowPos = abs (lfFont.lfHeight) + 175

 // Create a text font.
 oTextFont.CreatePointFont(120, "Arial", pDC);

 // Get the current text font height.
 oTextFont.GetLogFont(&lfFont);
 iTextHeight = abs(lfFont.lfHeight) + 10;
```

```
// Create a font for displaying column headings.
lfFont.lfWeight = 700;          // Make it bold, mormal is 400.
oColFont.CreateFontIndirect(&lfFont);
pDC->SelectObject(&oColFont);

// Compute the column spacings. Set the first column to 1/2 inch.
iRecNumPos = int(oDrawRect.Width() / 17);
iFoodIDPos = iRecNumPos + 50 + pDC->GetOutputTextExtent("##").cx;
iNamePos = iFoodIDPos + 50 + pDC->GetOutputTextExtent("XXX00000XXX").cx;
iPerishablePos = iNamePos + 50 + pDC->GetOutputTextExtent("Xxxxxxxxxxxx").cx;
iPricePos = iPerishablePos + 50 + pDC->GetOutputTextExtent("Perishable").cx;
iPurchasePos = iPricePos + 50 + pDC->GetOutputTextExtent("$00.00").cx;
iQuantityPos =iPurchasePos + 50 + pDC->GetOutputTextExtent("Purchase Date").cx;
iStoragePos = iQuantityPos + 50 + pDC->GetOutputTextExtent("Quantity").cx;

// Display the column headings.
pDC->TextOut(iRecNumPos, iRowPos,"#");
pDC->TextOut(iFoodIDPos, iRowPos,"Food ID");
pDC->TextOut(iNamePos, iRowPos,"Name");
pDC->TextOut(iPerishablePos, iRowPos,"Perishable");
pDC->TextOut(iPricePos, iRowPos, "Price");
```

```
pDC->TextOut(iPurchasePos, iRowPos, "Purchase Date");
pDC->TextOut(iQuantityPos, iRowPos, "Quantity");
pDC->TextOut(iStoragePos, iRowPos, "Storage Life");

// Create a space between the column heading and the text.
iRowPos += iTextHeight;
pDC->MoveTo(iRecNumPos, iRowPos);
pDC->LineTo(oDrawRect.Width() - iRecNumPos, iRowPos);
iRowPos += 20;

// Select our text font into the device context.
pDC->SelectObject(&oTextFont);

// Determine the row height.
iTextHeight = 20 + pDC->GetOutputTextExtent("Xy").cy;

// Print the records in a loop.
while (iRowCount<m_dataGrid1.GetVisibleRows())
{

//Display the current record number.
```

```
itoa(iRowCount,lpstrRow, 10);
cLine + lpstrRow;
pDC->TextOut(iRecNumPos, iRowPos, cLine);

// Print the data.
m_dataGrid1.SetRow(iRowCount - 1);
m_dataGrid1.SetCol(0);
pDC->TextOut(iFoodIDPos, iRowPos, m_dataGrid1.GetText());
m_dataGrid1.SetCol(1);
pDC->TextOut(iNamePos, iRowPos,m_dataGrid1.GetText());
m_dataGrid1.SetCol(2);
if (m_dataGrid1 GetText() == "-1")
      pDC->TextOut(iPerishablePos, iRowPos, "Yes");
else
      pDC->TextOut(iPerishablePos, iRowPos, "No");
m_dataGrid1.SetCol(6);
pDC->TextOut(iPricePos, iRowPos, "$" + (m_dataGrid1 GetText());
m_dataGrid1.SetCol(4);
pDC->TextOut(iPurchasePos, iRowPos, m_dataGrid1 GetText());
m_dataGrid1.SetCol(5) ;
pDC->TextOut(iQuantityPos, iRowPos, m_dataGrid1 GetText());
```

```
        m_dataGrid1.SetCol(3);
        pDC->TextOut(iStoragePos, iRowPos, m_dataGrid1 GetText());

        // Advance the row.
        iRowPos += iTextHeight;
        iRowCount ++;
      }

      // Perform the default action
      COleDBRecordView::OnPrint(pDC, pInfo);
    }
```

这看上去代码可真不少，也确实如此。处理打印输出是件相当困难的工作。但是，如果将任务分成几个小块，就不难看出该怎么做了。

需要了解的第一件事是 CDC **类的神秘的** pDC **对象。这就是我在前面提到的设备上下文。如果把** pDC **看作是调色板，也就是内存中绘制想要发送给打印机的信息的地方，会很有帮助。事实上，**Microsoft **已尽其所能在函数的命名和处理** GDI **的结构中孕育了这个观点。**

我们应该做的第一件事（除了产生似乎有成千上万个变量之外）是创建一些绘图工具。如果没有必需的绘图工具，就不能在调色板上绘图。在这里，我要说明如何创建三个基本的绘图工具，你会在大多数程序中用到它：刷子、画笔和字体。画笔可以是任何颜色，任何宽度，你甚至可以选择像点这样的绘图

图案。用画笔可以画线，它也包括了对象的轮廓。例如，矩形不需要轮廓，但大多数其它对象都需要。你可以画出一个只用填充色填充的矩形。我们来看一看刷子。刷子为实线对象提供填充色。刷子还可以有特别的颜色，你可以选择创建带图案的刷子，如交叉阴影线。最后，字体用来在屏幕上写出一些东西 。它有很多的特性，但大多数特性对典型的程序员来说是比较陌生的。程序列表 7.3 中的源代码显示了创建字体的几种方式，使用它们并不要求全面的桌面排版知识。

注 Windows 支持三种基本的绘图工具：字体、画笔和刷子。

仅仅有了绘图工具并不表示就可以使用它。画家可能在墙上挂了一支笔，但并不表示就可以用它来画画。同样地，当你在 Windows 中绘图时，首先必须选择工具。这就是 SelectObject()函数要完成的任务。它让你为绘图工作选择一个对象。

注 在使用工具之前必须首先选择工具。

对 Windows 中的每个设备上下文来说，都有一个刷子、画笔和字体对象。可以把它看成是有三只手，每只手都以不同的工具装备起来。在使用另一种工具之前，必须把当前的工具放下，然后选择新的工具。Windows 自动为你处理放下工具的任务。每次调用 SelectObject()函数时，它就放下旧工具并捡起新的工具。

每一个工具在很多重要的方面都很独特。如果想用 Arial 10 点字体，就需要一种特殊的字体工具。如果需要该字体的粗体格式，或者突然想以 Times New Roman 12 字体显示文字，就必须创建另一种字体工具。幸运的是，不必为想要

使用的每一种颜色都创建分离的字体——这个特征由设备上下文来控制。说明工具的独特性的另一个示例是画笔。你要为想要的每一种宽度和风格的线创建画笔。刷子在有些方面显得更为独特。可以用相同的函数创建想要的任何字体或画笔，但对每种刷子来说，就要调用不同的函数。例如，如果你想创建交叉阴影线刷子，就要在代码中调用CreateHatchBrush()函数而不是CreateSolidBrush()函数。认识到现有工具的局限性并按需要创建新的工具是很重要的。

注释 本示例的绘图命令做了打印机优化，分辨率范围是每英寸300到360点。如果你的打印机分辨率更高的话，可能需要修改位置值以获得清晰的画面。但是，不论打印机的分辨率有多高，你都可以看到这些命令是如何相互配合、从而为数据库应用程序提供某种形式的输出的。

在绘图方面我们为本例做的第一件事是，用Ellipse()函数创建一个椭圆。和大多数实线绘图命令一样，要指定图形左上角和右下角的坐标。Windows假定你想使用当前选定的刷子和画笔，所以无需指定这些值。

绘制椭圆之后，代码会在其中放入一些文本。我们要做的第一件事是，把背景绘图模式设置成透明方式，这样我们就不用擦去椭圆的任何部分了。另一种模式是不透明的，意思是在前景中添加某种绘图的同时，要替换背景的颜色。现在看看代码是如何用GetOutputTextExtent()函数来确定文本的长度的，其单位是像素。这大概是方便地确定一行文本行长度的唯一方式。确定了文本长度后，你就可以把这些文本在打印页面上居中放置了。下一行用TextOut()函数向打印机发送文本。本节代码的最后一行用SetBkMode()函数把背景绘图模式复位成不透明方式（缺省设置）。

现在讨论一下字体。创建字体有两种便利的方法，下一段中的代码对此做了说明。第一种方法是用 CreatePointFont()函数。你需要提供字体名称和想要创建的字体的大小，字体的单位是十分之一点。所以，如果你想创建 10 点的字体，就要指定字体大小值为 100。

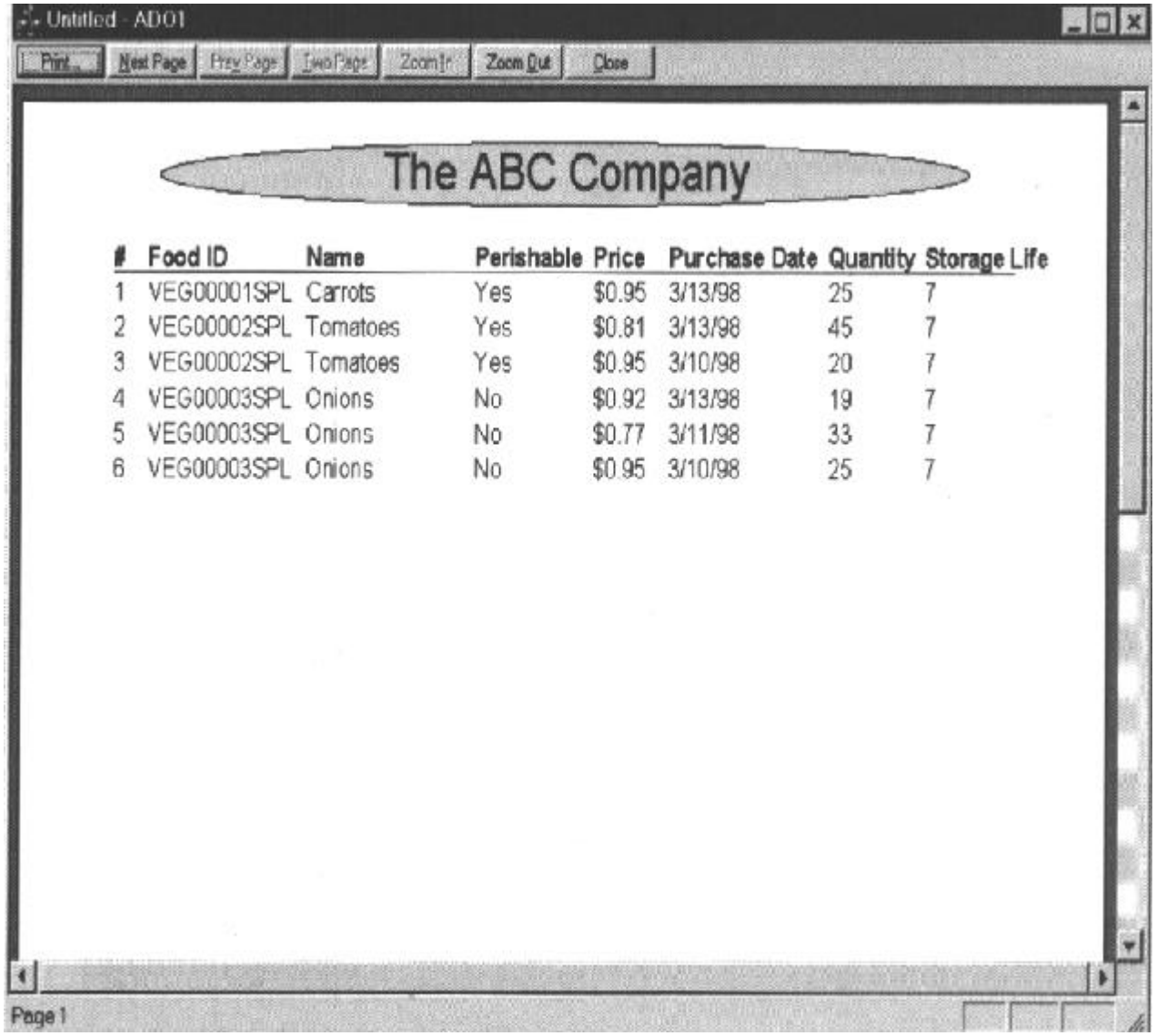
字体使用特殊的 LOGFONT 结构传送它们所需的全部参数。如果你使用了 CreatePointFont()函数，Windows 就会为你创建这个结构，并在创建过程中设置一些缺省值。如果不想要这些缺省值时该怎么办呢？我们可以使用 GreatePointFont()创建简单的字体（就像我们刚做的那样），然后用 GetLogFont()函数填充 LOGFONT 结构。现在，你已经有了功能完整的结构，可以稍微做一些修改。我之所以强调“稍微”是因为，如果你试图在 Windows 刚刚返回的结构中做大的更改，那是搬起石头砸自己的脚。我把字体的磅数从正常改成粗体，只要字体磅数没有太大的差别，这个修改就是可以接受的。现在可以用 CreateFontIndirect()函数创建字体的粗体风格。下面的工作只需要把 Windows 提供的、修改了 IfWeight 成员的 LOGFONT 结构传送回去也就可以了。

请注意，一旦代码把字体选进了设备上下文，就用粗体字体创建了一组报表头。请始终谨记，首先要创建工具，然后根据在设备上下文上绘图的需要选择工具。我们还创建了一组变量，用来跟踪列的定位信息。两个报表头和细节信息使用了这些定位变量。请注意，用来导出定位信息的方法和设备无关。换言之，不论你是用 300 dpi 的打印机还是 600 dpi 的打印机，它都应该看上去差不多。当然，这里的随机成分是，我们用当前字体提供的文本所占用的空间量，Windows 到底安排得怎么样。在大多数情况下，你会发现这个算法是精确的，

但是要在公司里所有打印机上都打印出精美的报表，还是需要下一些功夫的。

下一步在标题下面绘制一条线段。请注意，绘制线段的过程分为两个步骤。第一，用 MoveTo()函数把画笔移到打印页上的特定点，然后用 LineTo()函数实际绘制这条线。可以把 MoveTo()看成是抬起画笔的移动，把 LineTo()看成是按下画笔的移动。

打印例程的最后一部分使用在 DataGrid1 中已有的数据。该代码完成的工作是在数据网格中的单元间移动，并在打印机或屏幕上打印其内容。那么，我们的应用程序的打印输出看起来是什么样的呢？下图就是个打印页面示例。


Untitled - AD01
Print...
Next Page
Prev Page
Two Page
Zoom In
Zoom Out
Close
The ABC Company
Food ID Name Perishable Price Purchase Date Quantity Storage Life
1 VEG00001SPL Carrots Yes $0.95 3/13/98 25 7
2 VEG00002SPL Tomatoes Yes $0.81 3/13/98 45 7
3 VEG00002SPL Tomatoes Yes $0.95 3/10/98 20 7
4 VEG00003SPL Onions No $0.92 3/13/98 19 7
5 VEG00003SPL Onions No $0.77 3/11/98 33 7
6 VEG00003SPL Onions No $0.95 3/10/98 25 7
Page 1

第三部分

Visual C++与Internet



返回总目录

目　　录

第 8 章　设计 Web 页 4

8.1　HTML 概要 9

8.2　创建一个简单的 HTML 文档 15

8.3　ActiveX 适用的场合 47

8.4　COM+：Internet 的未来 51

8.5　从 Internet 上下载 ActiveX 54

8.6　增强 HTML 的功能 61

8.7　使用 DIANTZ.EXE 创建组件下载(CAB)文件 88

第 9 章　JavaScript 概览 95

9.1　ActiveX Scripting 概要 98

9.2　Java Scripting 基础知识 107

9.3　使用独立的脚本 115

9.4　使用 ActiveX 控件 126

第 10 章　创建 ActiveX 控件 138

10.1　理解 ActiveX 控件的一些背景知识 141

10.2 一个基于 MFC 的基本按钮编程示例 149
10.3 在 Netscape Navigator 和 Internet Explorer 206
中使用 ActiveX 控件 206
第 11 章 创建 ActiveX 文档应用程序 214
11.1 什么是 ActiveDocument（OLE 文档对象） 217
11.2 ActiveDocument 结构概述 236
11.3 创建 ActiveDocument 251
第 12 章 使用 URL 和 Moniker 270
12.1 URL Moniker 概述 271
12.2 创建 URL Moniker 276
12.3 超链接基础 278
12.4 理解超链接接口 283
12.5 使用 URL Moniker 285
第 13 章 使用 Internet 信息服务器(IIS) 302
13.1 ISAPI 概述 306
13.2 创建 ISAPI 扩展 315
13.3 在 Web 页中使用 ISAPI 扩展 324
13.4 创建 ISAPI 过滤器 331
13.5 使用 ISAPI 过滤器划分 Web 站点的密级 339
13.6 使用 ISAPI 扩展转发服务器信息 343
13.7 使用 ISAPI 过滤器请求用户名和口令字 355

第8章　设计Web页

以任何方式接触Internet都需要在一定程度上了解HTML（超文本标记语言）知识。对于这一点，无论你怎样努力，都不能躲开它。以任何形式正常进入Internet，都要涉及到访问Web页，在你领悟到这一点之前，这种说法看起来好像太笼统。在键入http:时，就会发现你自己正在进入Web页，这种Web页通过HTTP（超文本传输协议）使用了HTML，而不键入这些字母，就不能进行任何信息交换。

技巧　如果愿意的话，大多数情况下可以避免学习HTML标记(tags)，诸如Visual InterDev之类的产品使你很容易地创建含有ActiveX控件的Web页面。当然，这就意味着，为了创建并测试ActiveX控件，在购买Visual C++时不得不将另一个附加产品买回来。显然，使用诸如Visual InterDev之类的产品会使你受益，而购买并学习另一个产品则对你产生不利影响，根据你实际要创建的Web页数量，你需要对此权衡利弊。如果仅仅是偶尔地创建Web页，那么，手工写出代码可能仍是上策；反之，如果大部分时间你都是在创建Web页，那么就应该投资购买诸如Visual InterDev之类的产品了。Osborne/McGraw-hill公司将出版我的新著《Visual工作室：参考手册全集》(Visual Studio: The Complete Reference)，从这本书中你可以就怎样使用

Visual C++及诸如Visual InterDev之类产品，学到更多的东西，这本书介绍了怎样使用 Visual 工作室包含的各种语言产品在大环境下（特别是在团队环境下工作时）去创建应用程序。

作为对使用 ActiveX 感兴趣的 Visual C++程序员，你在创建 Web 页方面可能会花费大量时间，也可能不在这方面耗费时间，但是，你需要了解它们是如何工作的，以便于你能帮助他人使用你的控件。学习 HTML 起初看起来有点难，有大量的标记（就象程序语句一样）可以用于创建 Web 页。查看这些标记不成问题——市场上的任何浏览器都提供了 View/Source 命令，用它们就可以看到 Web 站点的任何 Web 页的原始 HTML。图 8.1 是一个典型例子。

注释 和本书中其它图示一样，对浏览器的配置不一样，可能看到的显示画面外观会有所变化。另外，Microsoft 和 Netscape 经常会改变它们各自的 Web 站点和浏览器的画面外观。最后，增加插件也会稍稍改变浏览器的外观。请记住，把本章中所有的图示及说明看作是你将看到的画面的一个基础，你的实际的屏幕显示可能会与本书所说明的有所不同。

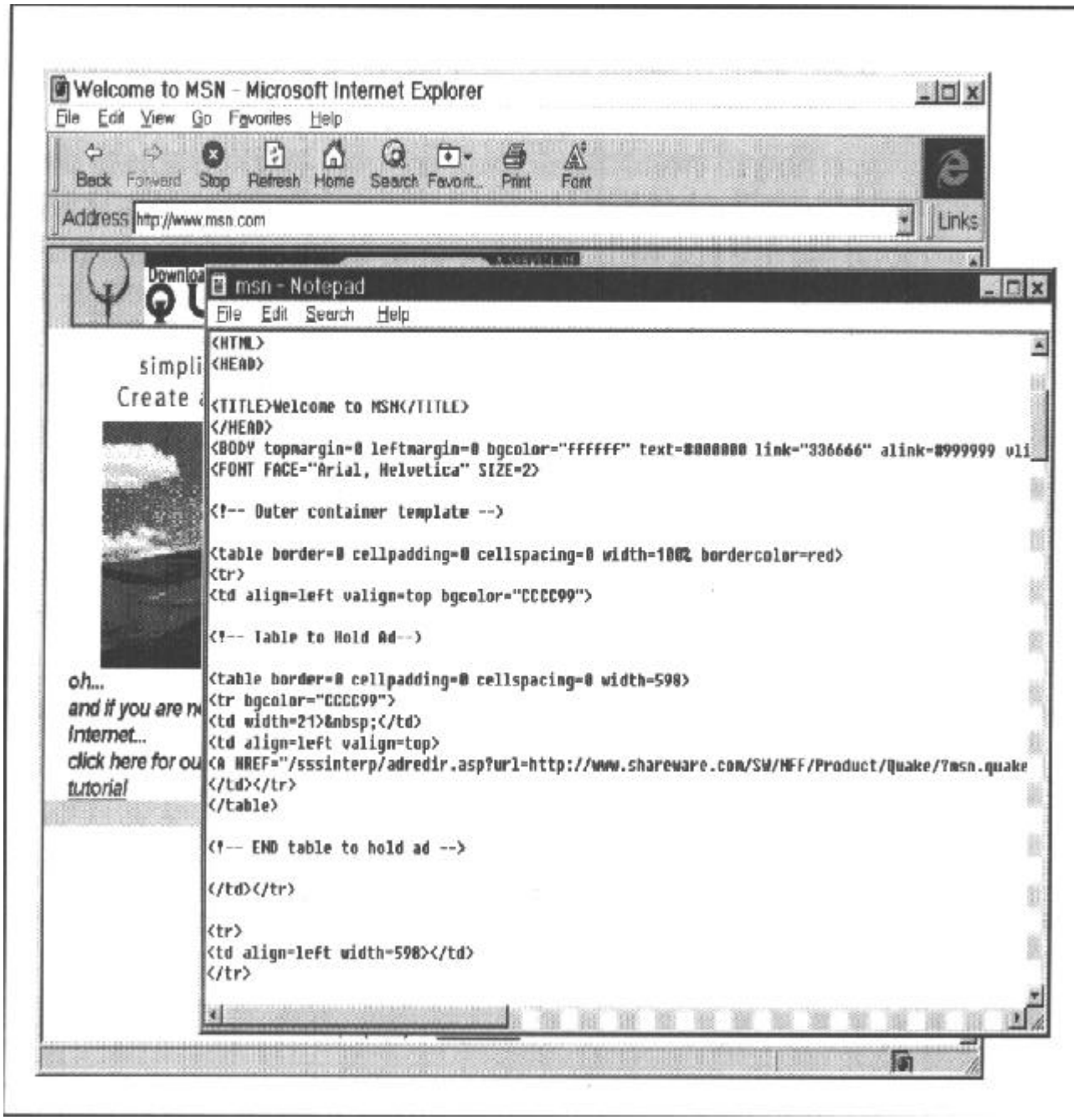


图 8.1　**你能够查看大多数** Web **页面的源代码——当你想学点新东西时这是个方便的特色**

至少有一种简单的途径开始学习过程，这正是我们这儿将要讨论的问题。仅仅使用 12 种 HTML 标记就可以构成一个 Web 站点，以后再用另外的标记对它们进行修饰。这就是本章第一节的内容。我们考察这 12 种标记，在任何 Internet 站点——甚至是提供最小级别功能的站点——都可看到它们。

技巧 如果要设计大量的 Web 页，那么，能利用 View/Source 命令而获益。先简单地找到一个含有你想使用的那种特色的 Web 站点，然后查看其基础源代码，从而知道别的程序员是如何完成这一任务的。在其它许多编程情况中，代码或被隐蔽，或被加密，或用其它不可访问形式。而 HTML 却保留了自己的类似英语的语法结构，即便是在一个 Internet 站点使用时也是如此。

一旦记住了这 12 种标记，我们就看一下需要使 ActiveX 工作的另外一个标记。我们将把你对一个 ActiveX 控件知道的每一样东西组合起来，产生一个非常简单的 Web 页（第 11 章将说明怎样创建这个 ActiveX 控件），然后，使用这个 Web 页就能干点事了，甚至可以把它作为以后创建新的 Web 页的模板。我的想法就是提供 HTML 的基本知识，而不是使你成为一名 HTML 编程权威者。

注释 本章不打算把需要知道的有关 HTML 的每一个细节都作介绍。有些人抱怨，就这个题目即便写一大本书也不足以完成这项工作。我们想做的是在一个非常基本的级别上——大多数同类书籍中漏掉的级别——考察一下 HTML，看完本章，就有了足够多的信息，使得学习其它 HTML 标记更加容易。

ASCII 文本文件能包含足够多的使 ActiveX 工作的信息，这看起来有些奇怪。学习了 ActiveX-专用标记之后，我们将研究一下浏览器是如何解释这一标记并使 ActiveX 工作的了。一旦知晓了 Microsoft 在 Internet Explorer3.x 和 4.x 中包含的小秘密后，就会发现完成上述任务并不是一件难事。

Microsoft 还引入了称之为 COM+（组件对象模型 Plus）的新技术。Microsoft 正是最终利用这一技术在 Internet 上扩大编程的。它不是 ActiveX 的替代物，ActiveX 仍保持作为创建组件的选择技术，但 COM+将最终会在 Internet 上取代 ActiveX。本章后面的一节将提供这一激动人心的新技术的概况，从而认识到在将来它会怎样对你有所帮助。

注 COM+最终会在 Internet 上取代 ActiveX，而 ActiveX 将仍保留作为 Microsoft 的应用程序组件技术。

掌握了基本的 HTML 标记和 ActiveX 标记之后，我们就对简单的 HTML 页进行两个容易完成的增强，这两个增强，一个是声音，一个是动画 GIF，它们摇旗呐喊，避免了枯燥。重要的一点是为你的 Web 站点增加情趣时，别把它搞得太大，以至于不能在合适的时间内下载。

本章的最后一节，讨论 ActiveX SDK 提供的一个实用程序：DIANTZ.EXE。

它允许压缩一个或多个 ActiveX 控件，减少下载时间。Microsoft 正是使用这个实用程序产生了用于发布 Windows95 的 CAB 文件，可以发现，DIANTZ 实用程序的功能远远不止是用于 ActiveX 开发。

8.1 HTML 概要

关于 HTML，即使写上一大厚本书，也不足以涵盖所有标记的全部变化。让我们首先讨论一下“标记”这个术语。每一个 HTML 语句由一个或多个标记组成，一个标记就象一个程序设计语句。有的标记用于标明 HTML 文档各个章节的开始与结束，而其它的标记则用于标明进行其它处理。由于公司更多的涉足 Internet，标记的复杂性也在变化，我们将讨论落入本章范围的两种变化，但是，由于篇幅所限，我们将不讨论其他的变化（本章将指出在什么地方可以找到一些另外的信息）。

注　标记类似于微编程语句，允许格式化 HTML 文档或者处理信息。

最基本的 HTML 文档大概包含 12 种不同的标记，其中一些是强制性使用的，另一些仅用于提供指定类型的功能。我们仅讨论标记的最简单形式——但能够对大多数标记添加限定词以影响它们工作的方式，例如：可以添加一个字体描述，或者让标记对其所在页的文本进行居中处理。表 8.1 是将要讨论的 12 种基本标记的概要。

注　据经验估算，包括图形在内，Web 页要保持小于 60KB，否则，下载会占用太多的时间。

表 8.1　12 种基本 HTML 标注概要

标记	使用方式	说明
<HTML> 和 </HTML>	强制使用	每个 HTML 文档都以<HTML>开始，以</HTML>结束，这样浏览器就可以知道从什么地方开始和结束读取
<HEAD> 和 </HEAD>	正常使用	当查看 HTML 文档时，你将看到一个头部和一个正文，头部通常标识 Web 站点并定义页面设置
<BODY> 和 </BODY>	正常使用	将把 Web 页内容的主要部分放在这两个标记之间，正文部分通常包含着访问站点的用户要查找的信息
<Hx>和</Hx>	选择使用	书籍和其它文本形式，通常把它们所包含的信息，用标题分成章节，本标记允许对 Web 站点添加标题
<P>	选择使用	HTML 总是假定：你键入的所有文本，不论它们是否在一行，都是同一段的一部分。使用本标记增加一个回车符和两个换行符（标识段结束）
 	选择使用	常常想要在不增加新行的情况下回到页的左端，本标记可以完成这一工作，它添加一个回车符和一个换行符

续表

<HR>	选择使用	水平标尺标记允许一行可以跨页，它还在段落之间增加空白
和	选择使用	与其它文档的超链接是 Web 页最普通的组成部分
和 	选择使用	锚地标记允许从页的一点移动到另一点。超级链接使用来访问这一标记
和	选择使用	不同的浏览器对强调标注反应不同，大多数浏览器都是用斜体显示文本
和 	选择使用	和强调标注一样，本标记由浏览器决定如何对标记作出反应。大多数浏览器看到这一标记时用黑体显示文本
<PRE>和</PRE>	选择使用	经常不想让浏览器对文本重新格式化，预格式化标记指示浏览器保持文本格式不变

现在已经对这些标记的作用有了基本了解，让我们再对它们中的几个进行详细的讨论。下面的章节讨论一些标记，它们会使你在多少有些一致的基础上进行使用，并且在大多数情况下使用时要做一些工作。不必担心在讨论中不能完全掌握它们，在“创建简单的 HTML 文档”那一节中会详细地讨论实际使用过程。了解标记的工作方式，有助于你弄明白它们与浏览器是如何相互作用的。

理解<HEAD>标记

头部让浏览器在开始显示时知道你对该页做些什么，但它本身并不提供任何形式的实际内容。例如，请看图 8.2 中 http://www.microsoft.com 的源代码。

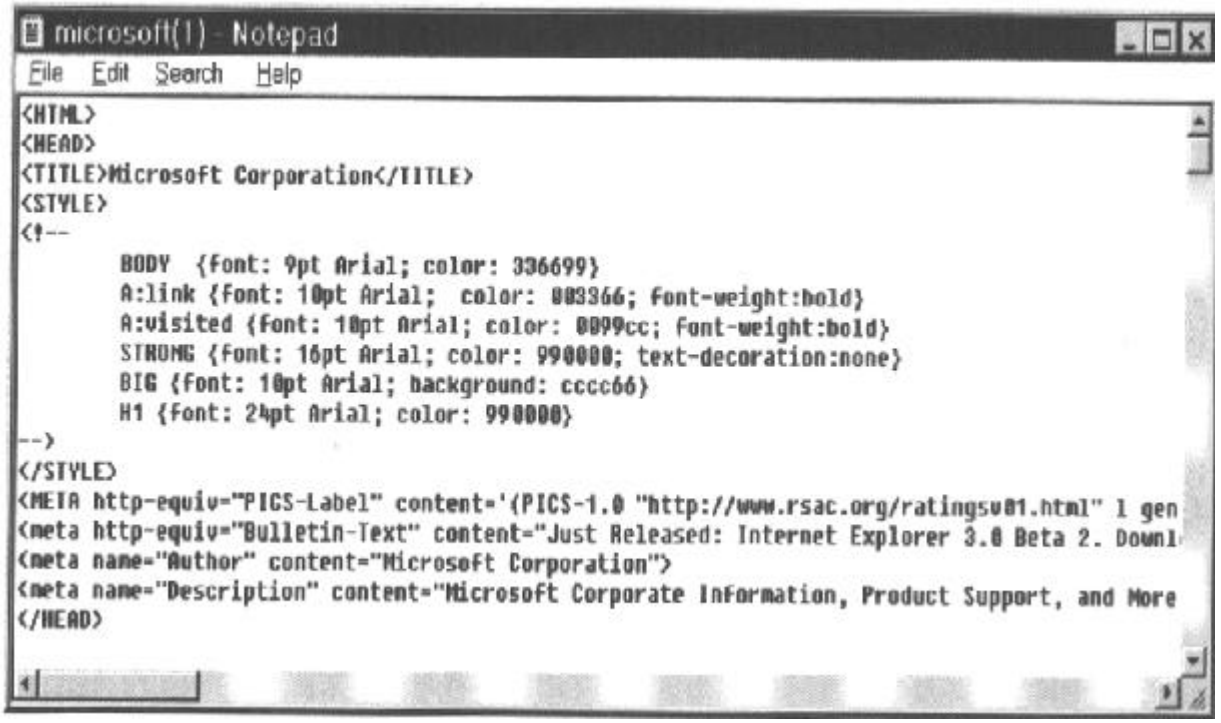


图 8.2 <HEAD>标记不影响页的内容，但确实定义了页的外观

注意到头部会有一个标题，一些体例命令和几个元命令。事实上，这三个元素代表了通常可以在 HTML 页头部中找到的仅有的三种标记。

不要以为<META>标记是在头部部分找到的脚本的唯一形式。例如，如果访问 http://www.netscape.com，就可以在其头部找到 JavaScript 命令。不用在乎页设计者使用哪种脚本语言，关键是脚本的作用是为页的某些元素添加指令。

一种方法是把头部当作页的其余部分的定义区。它影响的是页的整体外观而不提供任何形式的内容。事实上，通常总是想对访问站点的访问者隐藏掉这一部分。

使用<Hx>标记添加标题

与一本书中的标题把文本分隔成小节一样，HTML 文档中的标题标记（<Hx>）把内容部分分隔成不同部分。HTML 规范提供了 6 个级别的标题，通常，这 6 个级别间的唯一差别是显示标题时使用的字体大小。标题的审美需求是人们所要求的，使用标题并无什么神秘可言，它们仅在文本的小节之间创建物理分隔，这有助于访问你的站点的人理解你所提供的材料。

技巧 数字小的标题级别使用的字体，比数字大的标题级别的使用的字体大。例如，<H1>比<H2>使用户可以看到更大的字体。

怎样创建一个小节标题呢? 在<Hx>标记中，简单地用 1 到 6 之中的一个数字代替 x，然后键入标题文本，之后在行尾加上</Hx>。例：

```
<H1>This is b heading </H1>
```

要记住的另一点是，不必在标题尾部为标题添加诸如<P>，
或<HR> 之类的行结束标记，这是因为结束标记</Hx>自动地添加了所需数量的空白，当然，这并不阻止你加入所需的其它额外空白。

链接与锚地

链接与锚地提供了访问一个 Web 站点的特殊手段。每次访问一个 Web 站点时，总会看到一些文本加了下划线，这时大概就是看到了一个链接。一个链接把当前位置与某个另外的位置相连接。事实上，大多数 Web 页使用最常见的链接形式。第一种是创建到另一页的链接（用 Internet 术语来说是一个 URL），这种形式更常见； 另一种是创建到当前页的另一小节或到另一页的某一指定部分的链接，使用它可以找到一个锚地（一会儿会讨论它）。下面是两种常见的链接形式的例子，注意到它们都在<A> 标记中使用了 HREF 关键字。

```
This is a <A HREF="http://www.microsoft.com">Document</A>Link

This is a <A HREF="http://www.microsoft.com#MyAnchor">Anchor</A>Link
```

注 链接将一页与另一页相连接；锚地创建到一页中指定位置的一种连接。

在这两种情形中，标记<A>与标记间出现的文本都是 Web 站点上加了下划线的句子的一部分。在<A>与之间应该总是要放上某些文本，否则你的链接对于用户是不可见的！注意到这两种链接看起来几乎完全一样，但有一个差别，那就是锚地链接使用#号将 URL 与锚地名分割开，上例中，使用 MyAnchor 作为锚地名，第二种链接首先处理 URL，然后在向用户显示内容之前，先处理页上指定的锚地位置。

技巧 若创建到当前页锚地的一个锚地链接，则不必提供<A>标记中的 URL 部分，可缩写为。

锚地链接要求在<A> 标记中指定页的某个地方创建锚地（除非是在当前页

找一个指定位置，这时不必提供URL）。例如，我们看到了一个到http: //www.microsoft.com 上的 MyAnchor 的锚地链接，但是这一位置的锚地看起来是什么样的?

This is the Anchor<A> for this page.

如你所见，锚地和链接之间的差别是 NAME 关键字，锚地总是用 NAME 代替 HREF。另外，大部分浏览器都不对<A>与之间的文本作高亮或加下划线处理，这是因为确实不用它们做任何事。

8.2 创建一个简单的 HTML 文档

在前面的小节中，我们讨论了创建HTML 页所用的12 种基本标记。每个人都可以按这种方式使用这些标记。现在，该讨论一下怎样在 Web 站点上使用这些标记创建一个页了。清单 8-1 列出了一个简单 Web 页的源代码。显然，决不会在实际应用程序中创建这样（简单的）一个页，这一页的全部目的在于说明12 种标记是如何工作的。

程序列表 8.1

```
<HTML>
<HEAD>
<TITLE> Sample HTML Page </TITLE>
</HEAD>
```

<BODY>

<H1>This is a Level 1 heading。 </H1>

Notice that the "Level 1"portion of the heading is emphasized using the

EM tag.<P>

We ended the previous paragraph with a < STRONG> P< /STRONG> tag.

This paragraph will end with a BRtag.

Notice how we used the STRONG tag to add bold text to

the previous paragraph.<HR>

Horizontal rules also have a place on Web sites. You'll use them most

often to provide separations between major areas of text . Of course,

most Web sites are starting to use frames because they're more flexible .<HR>

Here are the other five levels of headings.<P>

<H2>Level 2</H2>

<H3>Level 3</H3>

<H4>Level 4</H4>

<H5>Level 5</H5>

<H6>Level 6</H6><HR>

Let's look at some other HTML tags.<P>

Click Here to display the list page.

Click Here to display the graphics page.

Click Here to display a page with tables.


```
Click <A HREF="FORM.HTM">Here</A> to display a page with forms.<BR>
Click <A HREF="#MyAnchor">Here</A> to see another area of this page.<P><P><P><P><P>
This is an <A NAME="MyAnchor">Anchor</A> link area.
</BODY>
</HTML>
```

这些源码初看起来有些唬人，但这里没有任何前面小节中没有讲到的东西。首先，第一个标记< HTML> 定义了页的开始，在源代码程序列表的末尾可以看到它的配对符号</HTML>。其次，我们把文档用<HEAD>及<BODY> （及其配对符号）将文本分成了两部分。唯一没见过的标记是<TITLE>，它定义了浏览器标题栏中的标题，这一标记并不是绝对必要的，但给 Web 页一个标题很好，使其它人知道它们在看什么。另外，标题还可以帮助在自己的页间航行以找到出问题的部分。

注 <TITLE>标记在浏览器标题栏中显示你的 Web 页的名称。

页的正文部分以一个 1 级标题开始，接着是加了特殊属性的一些文本，其后是一条水平标尺标记，在这里，还可以看到能表明<P>与
标记有什么差别的一些文字，过一会儿就会看到它们是怎么一回事。

程序列表的下一节说明了标题的其它五个级别的差别，大多数浏览器只是简单地使用不同的字号来改变标题的级别。不幸地是，大多数浏览器没有足够多的可读的字体大小来对应全部标题级别。可能需要避免使用第 6 级标题，因为它们太小，以至于某些人看不清。第 5 级标题已经是相当小了。

在这个示例的最后一小节含有一些链接，现在还不能看到链接上去的页，这仅是后面章节的一个练习。但是，请看第五个链接，可以注意到它是当前页的一个锚地。正文部分的最后一个工作行包含着这个锚地。一定要花些时间在你的浏览器上看一看它们是如何工作的。

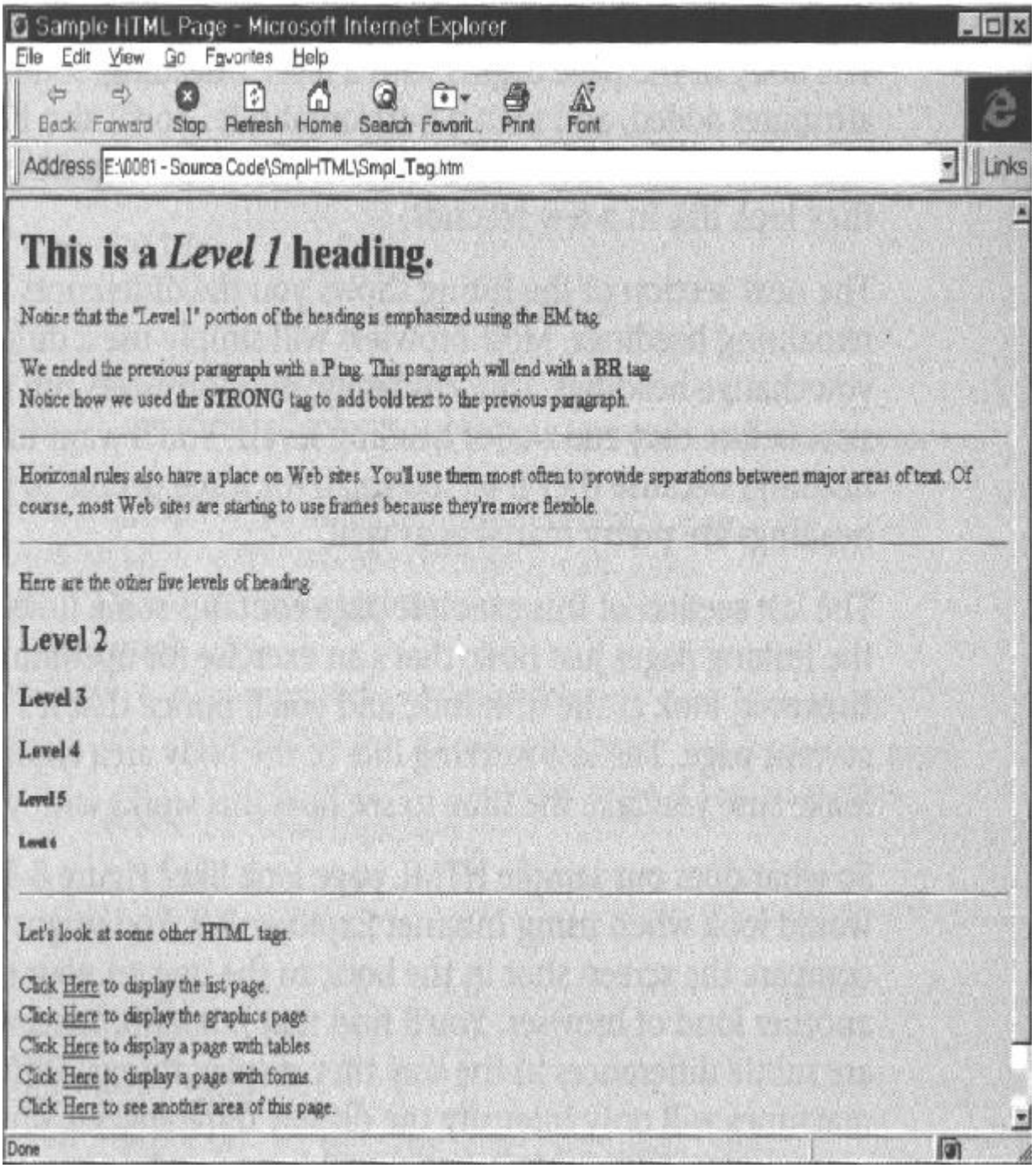


图 8.3　HTML **页的样本，用以说明在** Internet Explorer 3.0 **中** 12 **个基本标记的作用**

那么，我们的 HTML 样页看起来是什么样子呢? 图 8.3 是使用 Internet Explorer 3.0 时可以看到的样子。如果你使用其它浏览器，就要用些时间对书中列出的屏幕显示与你自己的屏幕进行比较了。可以发现，并不是所有的浏览器都是一样的，在显示东西时，它们会有一些细小的差别。机器的差别只能加剧显示的差别。考虑到这些都是常见标记，想象一下，如果使用了某个浏览器支持而其它浏览器不支持的一些更奇异的标记时会发生什么呢?!

现在到了扩充我们的 HTML 知识的时候了。下面几节将要说明一些其他标记，尽管在这些章节不准备讨论全部细节，但它们提供的足够信息，已经可以使你在访问其它 Web 站点并查看源码时能看懂它们。

使用列表

你看到过不包括任何形式的公告式列表或过程程序的技术书籍吗? 把复杂的思想用容易理解的语句片段表达出来，恐怕是作者们能够进行的最基本的任务之一。列表——是枚举想法的小段文本——是大多数情况下解决问题的答案。考虑到大多数 Web 页的大小限制，列表就变得更加重要。HTML 支持三种不同形式的列表：

- 无序列表　这种列表使用公告牌形式，以标记开始，并以 标记结束，在列表中的项以标记开头。
- 有序列表　过程或其它具有编号的文本形式考虑使用有序列表，这种列表格式使用和标记，和无序列表一样，每一项以标记开头。
- 术语汇编　这种特殊的列表使用由<DL>和</DL> 标记括起来的两类项元

素。被定义的项前辍以<DT>标记，而项的定义则前辍以<DD>标记。

尽管只有三种列表形式，但就如同其它程序设计方案中一样，可以用嵌套方法实现多种功效。讨论页的样本的代码时，就会看到列表 8.2 中是如何实现这一点的了。

注　总要对 HTML 页加入大量的注释，因为可能有不是程序员的人们来不时地编辑它们。

在我们讨论的三种形式的列表中，术语汇编是最灵活的。它使用的两种类型的列表元素使显示的文本看起来更美观。<DT>（项）标记和<DD>标记（定义）间的差别在于其中之一是被缩进的。可以用这一特色，作出与术语汇编类型元素显示无关的特殊功效。和本章中讨论的一些其它特殊文本标记一样，列表标记（，<DT>和<DD>）在行尾自动插入一个
标记。使用
标记使得文本变成下一行，但保持列表仍在一起。

现在又该看一下列表代码的例子了。程序列表 8.2 显示的是程序列表 8.1 中看到的第一页的第一个链接页。它包含着三种基本列表形式，以及一个说明怎样对它们进行嵌套的例子。请确保将产生的文件命名为 LISTS.HTM，这样才能使链接按所预期地那样工作。特别要注意这是第一个使用注释（<!-文本>标记）的页面。

程序列表 8.2

```
<!-本页目的是说明如何使用列表标注。->
<HTML>
<HEAD>
```

<TITLE>Creating Lists</TITLE>

</HEAD>

<BODY>

<H2>There are three different kinds of Lists:</H2>

<!-**显示无序列表**->

<H3>Unordered</H3>

Item 1

Item 2

<HR>

<!-**显示有序列表**->

<H3>Ordered</H3>

Item 1

Item 2

 <HR>

<!-**显示说明两种列表元素类型的术语汇编列表**->

<H3>Glossary</H3>

```
<DL>
<DT>A Term
<DD>this is the definition for it.
<DT>Another Term
<DD>Yet another definition.
</DL><HR>

<!-显示的是嵌套列表的一种方法->
<H3>Nested List</H3>
<OL>
<LI>Item 1
    <UL>
    <LI>Subitem 1
    <LI>Subitem 2
    </UL>
<LI>Item 2
</OL>
</BODY>
</HTML>
```

注　尽管有其它方式创建注释，但通常习惯于用<!－文本>标记。

你已很好地了解了怎样对各种列表标记写出代码了，现在就能够看到这一

页的样子了。图 8.4 显示的是怎样使用列表的一个典型例子，请注意，三种列表形式都没有怎么进行修饰，但它们却都正常有效。

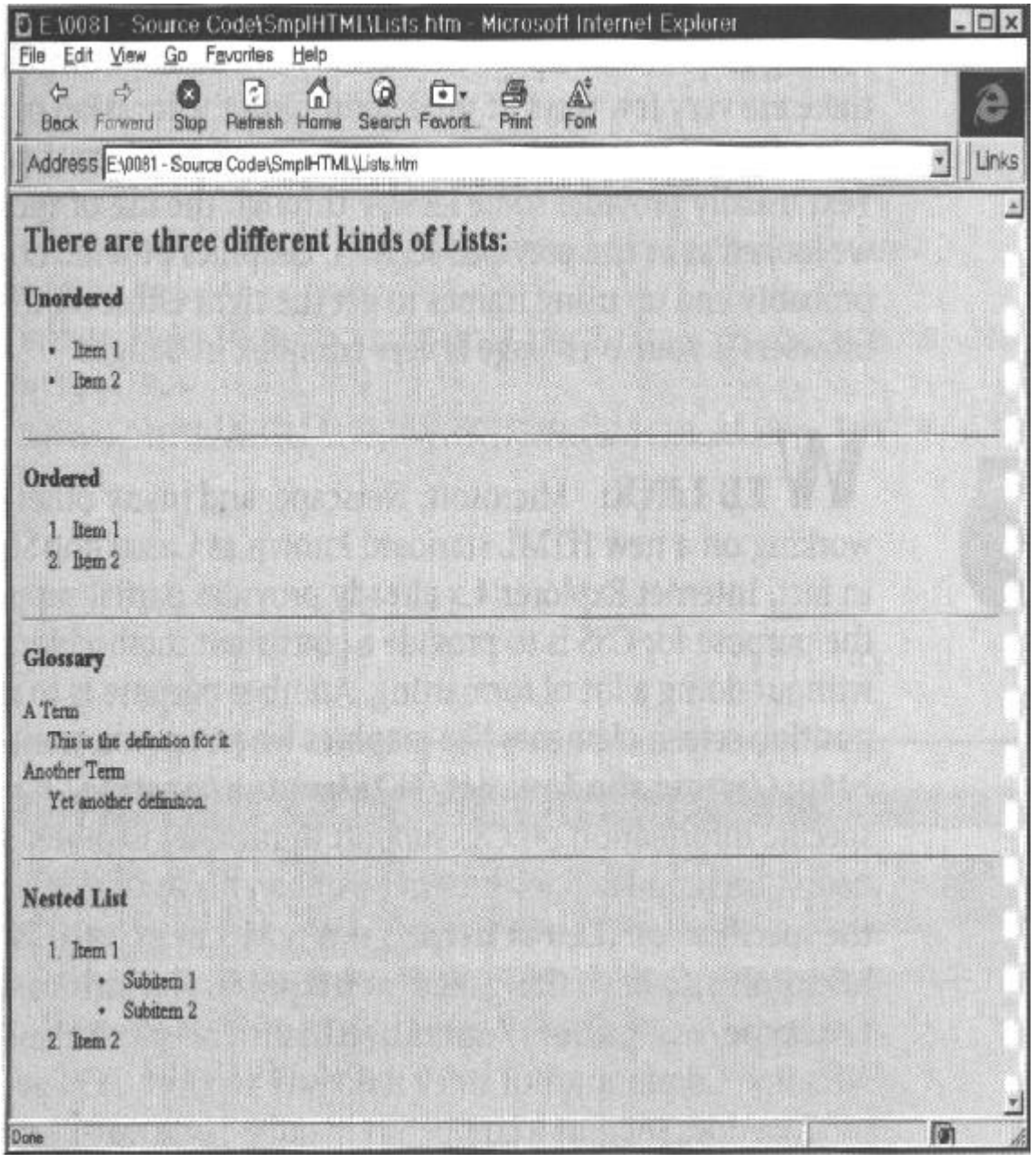


图 8.4　列表标记：容易地在 Web 页中枚举指定要点，或提供术语汇编类型的元素

添加图形

图形确实可以对文本页进行乔装打扮。我们大多数人都会同意这样的说法：如果作者在技术手册中添加了足够多的正确的图形图像，理解起来就会容易得多。也可以向 Web 页中加入图形，然而，添加的方式上却有一些限制。

技巧 一些人过分热衷于向它们的 Web 站点添加图形，导致了大多数人甚至不能看到这些图形的大部分，当用户在下载图形时，如果用了一分钟还不能完成下载，他们就会按下浏览器的 Stop（停止）按钮。事实上，一个良好的估算是使 Web 页的大小（包括图形和 ActiveX 控件）保持在 60KB 以下。

你要面对的最大限制之一是能够显示的图形种类，许多浏览器只处理 GIF（又称为 CompuServe 格式）和 JP（JPG）图像这两种格式——它们是由 HTML 支持的。你的浏览器上显示的其它类型的图像，都是使用了插件的结果。Netscape 和 Internet Explorer 都支持各种插件，例如，更流行的插件之一允许用浏览器观看 AVI（电影）文件。Web 页设计者的问题是不能指望别人都拥有这些插件。于是，如果计划创建一个受欢迎的 Internet 站点，那么大概就要坚持使用 GIF 和 JPEG 图像了。

注 使用 GIF 和 JPG 图像可以获得最大的兼容性，而在知道了用户使用什么浏览器时，使用其它图像类型则可以获得最大的美观效果。

显示图形图像是相当容易的，必须做的全部工作就是使用标记，这一标记使用 SRC 属性来接受一个文件名，例如：

把图形放入一个页面就不像你想像的那么容易了，像文本情况时一样，由于不能定义它们的确切位置，除了使用框架(frame)或表格之外，没有多少办法来预先确定图形的位置。通过使用诸如在前一小节中看到的列表标记技巧，文本通常提供了某些灵活性，而图形却不提供这样的灵活性。如果你设计中的 Web 页非常复杂，那么大概一直要使用框架(frame)来获取大部分浏览器的正确响应了。

Web 链接 Microsoft、Netscape 和其它供货商正在制定称之为层叠式电子表格（CSS）的 HTML 新标准。事实上，Internet Explorer 4.x 已部分地支持 CSS 。CSS 的一个目的是提供前后连贯的方法，从而不用做大量格式化工作就可以显示数据。另一个目的是，使诸如图形这样的屏幕元素的定位更容易。如果想找到 Internet Explorer3.x 支持的 CSS 的具体信息（写本书时，Internet Explorer 4.x 的 Web 站点还没有更新），那么请查看 http://www.shadow.net/%7Ebraden/nostyle/ 。可以在 http://www.w3.org/pub/WWW/Style/CSS/，找到规范本身。Microsoft 在 http://www.microsoft.com/turetype/css/gallery/entrance.htm 也有一个 CSS 展台，这一特定的站点不能用 Netscape Communicator 访问，因为到写本书时它还不支持 CSS。这一产品也支持称之为 Javascript Style Sheets 的竞争标准。

使用标记能够定义文本和图形混合显示的方式，所要做的一切工作

就是加入 ALIGN 属性，有三种方式对齐文本：

- ◆ ALIGN=TOP
- ◆ ALIGN=MIDDLE
- ◆ ALIGN=BOTTOM

注　当图形与文本相邻显示时，ALIGN 属性允许你改变文本的显示位置。

让我们看一下向某个 Web 站点加入图形的简单代码。在“使用窗体(FORM)”一节中，我们将更深入地考察图形的定位问题。程序列表 8.3 中的代码显示了图形图像，并说明怎样使用 ALIGN 属性（这一程序列表使用了我作的一个 GIF，你可以用手边的任何 GIF 文件代替它）。这种方法对于显示公司形象标志或在一个页上显示其它简单图形是适用的。图 8.5 则显示这些代码显示出的图像。

程序列表 8.3

```
<HTML>
<HEAD>
<TITLE>Using Graphics</TITLE>
</HEAD>
<BODY>
<H2>You can align text and graphics in one of three ways.</H2>
<H3>Top</H3>
<IMG SRC="ColorBlk.Gif",ALIGN=TOP>This is at the TOP.</HR>
<H3>Middle</H3>
<IMG SRC="ColorBlk.Gif",ALIGN=MIDDLE>This is in the MIDDLE.</HR>
```

<H3>Bottom</H3>

This is in the BOTTOM.</HR>

</BODY>

</HTML>

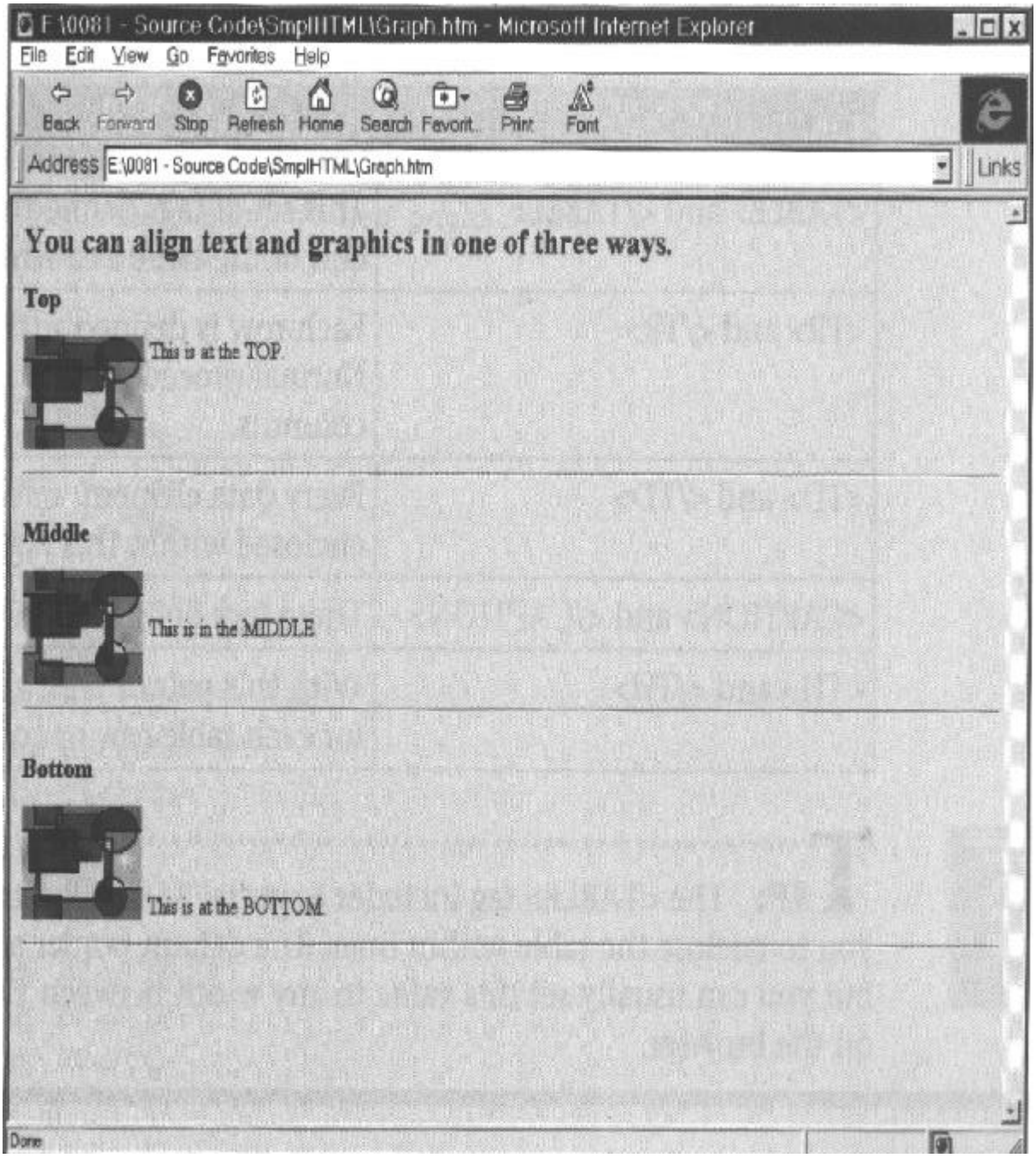


图 8.5　添加简单图形是相当容易的

Web 链接 你可能已经注意到某些人用图形而不是用文本来创建链接（详见本章前面一节“链接及锚地”）。这种技术称之为映射。如果没有实用程序的帮助，在 Web 页中使用映射很快就会糊涂起来。在 http://www,ecaeta.ohio-state.edu/tc/mt/ 上可以找到一个称为 Map This 的非常出色的共享实用程序，可以帮助你创建图形映射。（由于这个站点十分繁忙，也许会碰上麻烦，但坚持试下去，因为这个实用程序非常值得去努力得到它。）

创建表格

表格化数据差不多总是所有商务展示的一部分。电子表格帐页（spreadsheets）仅仅是会计师分类帐帐页在计算机中实现的一个例子。在 HTML 页上创建表格是有意义的，但是需要有一整套标记来完成这项工作。这里列出了所有必须的制表标记。

标记	说明
<TABLE>和</TABLE>	这两个标记定义表格的头和尾
<TR>和</TR>	用这两个标记定义每个行，正规步骤是先定义行，再定义列
<TD>和</TD>	每一个数据元素，或称做列，都用这一对标记括起来
<CAPTION>和</CAPTION>	这些标记定义表格的标题
<TH>和</TH>	使用这一对标记创建表格中每个行或每个列的标题

技巧 <TABLE>标记有一个特殊的 BORDER 属性，允许你用线条把表格围起来，缺省的边界（线条）值是宽度 1，也可以根据浏览器设置为 1 到 6 之间的任一个数字。

我们来看一份简单的代码，它定义了两种不同的表格。程序列表 8.4 说明了怎样使用本节谈到的那些简单标记来实现两种不同的效果。第一个表格包含的是简单文字元素，而第二个则以复杂的方式混合了文字和图形。

技巧 可用<CENTER>和</CENTER>这对标记对表格中的文本进行居中处理。这一对标记对放入文档的任何文本都可以起作用，而且它们对图形也起作用，请参考程序列表 8.4。

程序列表 8.4

```
<HTML>
<HEAD>
<TITLE>Creating Tables</TITLE>
</HEAD>
<BODY>

<!- 创建简单的文本表格 ->
<H3 Tables are an important part of Web pages.</H3>
<CAPTION> A Simple Text Table </CAPTION>
<TABLE BORDER>
    <TR>
```

```
        <TH></TH>
        <TH>Column A</TH>
        <TH>Column B</TH>
        <TH>Column C</TH>
</TR>
<TR>
        <TH>Row 1</TH>
        <TD>Entry A1</TD>
        <TD>Entry B1</TD>
        <TD>Entry C1</TD>
</TR>
<TR>
        <TH>Row 2</TH>
        <TD>Entry A2</TD>
        <TD>Entry B2</TD>
        <TD>Entry C2</TD>
</TR>
<TR>
        <TH>Row 3</TH>
        <TD>Entry A3</TD>
        <TD>Entry B3</TD>
        <TD>Entry C3</TD>
</TR>
```

```
</TABLE><HR>

<!-  创建图文混合的复杂表格  ->
<H3>You aren't Limited to text in a table.</H3>
<TABLE BODER=3>
    <TR>
        <TD><IMG SRC="ColorBlk.Gif"></TD>
        <TD><IMG SRC="ColorBlk.Gif" ALIGN=MIDDLE>Text and Graphics.</TD>
        <TD>Text Alone</TD>
    </TR>
    <TR>
        <TD>Some more text. </TD>
        <TD><CENTER><IMG SRC="ColorBlk.Gif"></CENTER></TD>
        <TD>Some more text. </TD>
    </TR>
    <TR>
        <TD><IMG SRC="ColorBlk.Gif"></TD>
        <TD><CENTER>Some more text .</CENTER></TD>
        <TD><IMG SRC="ColorBlk.Gif" ></TD>
    </TR>
</TABLE>
</BODY>
</HTML>
```

注 用缩进格式写代码可以使表格中的行和列分开，而且容易看清。

这些代码看起来真不少，手工写起来令人发烦（类似于这样的代码从另一角度说明，如果计划写出大量 HTML 页面代码的话，就应该多用些精力让 GUI 前端工具来帮你的忙）。图 8.6 是在浏览器看到的屏幕显示。上述代码还说明了一个要遵守的基本规则：工作时总是先行后列。如果遵守这一简单规则，在制表时就不会遇到什么麻烦。注意到第二个表格是图文混合的，尽管表格看起来有些复杂，但先行后列的同一原则仍然成立。请注意，第一个表格使用的 1 宽度边界与第二个表使用的 3 宽度边界有所差别，3 宽度边界占用更多的空间，但也更引人注目。

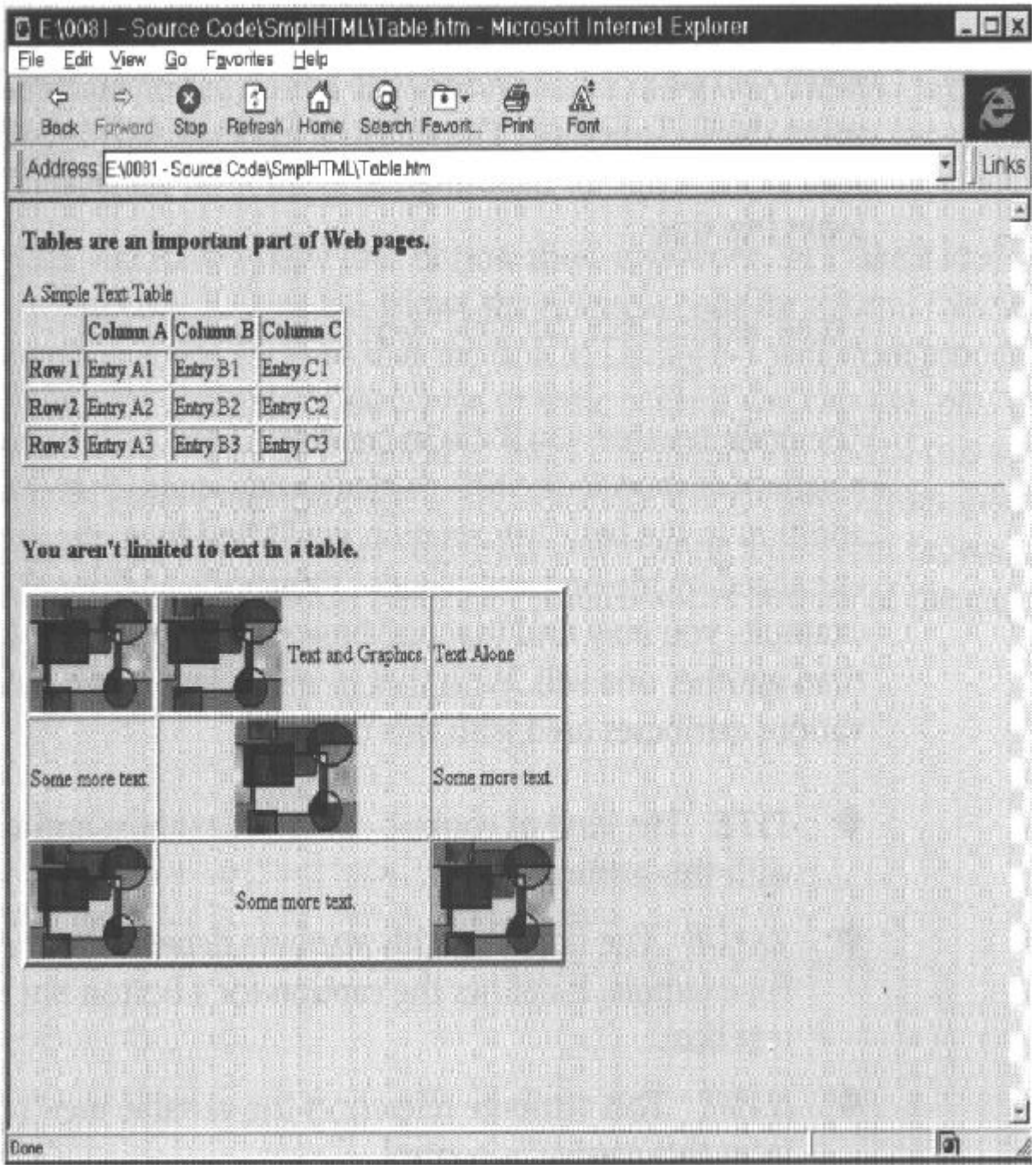


图 8.6　表格容易实现：先指定行，再指定列

第一个表格中行和列中都使用了<TH>。标准文本和表头文本有两个主要差别：第一，标准文本通常是左对齐的，而表头文本则是居中的。可像以程序列表 8.4 中那样，用<CENTER>来改变它们的行为。第二，表头文本使用粗体字。

注：用<CENTER>可以对 Web 页上任意位置的文本（包括表格和窗体中的文本）进行居中处理。

使用窗体

我们要讨论的最后一段基本的 HTML 代码是窗体，窗体有多种用途——大部分都是和数据项（Entry）打交道而不是用于数据传播（dissemination)。到目前为止，我们所看到的全部技术中，在使用标准的 HTML 向 Web 站点添加类似于单选按钮和复选框之类的项时，窗体提供了唯一的方法。因为 ActiveX 将从根本上大量地代替了使用窗体的需求，尽管了解窗体是使用 ActiveX 的另一种选择是重要的，但我们在这里并不打算介绍使用窗体的所有难以预测的变化。

每个窗体都以<FORM>标记开头。除了标记本身，还必须包括进两个属性：METHOD 和 ACTION。METHOD 属性定义了你打算怎样处理窗体收集数据的方式，这有两种标准的方法：POST 和 GET。POST 方法更多地用于数据项（Entry）窗体，它允许你从客户机向主机发送数据。GET 方法用于数据查询窗体。例如，当你要到你喜爱的 Web 搜索页时，实际上你就是要填一份使用 GET 方法的窗体。ACTION 属性则让 Web 服务器知道到哪里去找一个 CGI 脚本，用它处理来自窗体的输入。一个典型的<FORM>标记如下：

```
<FORM METHOD=GET ACTION="/cgi-bin/query">
```

创建一个窗体使用的主要标记是<INPUT>标记。有许多方式——多于这里介绍的——使用这一特定的标记。除了在这里可以找到的基本信息外，浏览器供货商通常对<INPUT>标记进行增强，使之提供附加功能。例如，你或许会发现某个浏览器比另一个浏览器支持更多种类的按钮。了解了这些后，让我们快速地看一下这一标记使用的各种属性吧。

TYPE 这一属性定义控件的类型——按钮、文本框等等。

VALUE 这一属性的效果取决于控件的类型。例如，它定义按钮的标题或文本框的内容。

NAME 这一属性指定引用该控件的变量名称。

SIZE 这一属性定义控件的大小——通常是按字符计算的宽度。

CHECKED 具有选中或未选中两种状态的单选按钮或其它控件使用这个属性。

ROWS 这一属性定义了为一个特定控件分配的行数——通常是控件的高度。

除了<INPUT>标记，还可以使用<SELECT>标记创建弹出式列表框或菜单，它工作的方式与本章前面创建的列表相同。要用</SELECT>标记结束列表框。在<SELECT>和</SELECT>标记之间，使用<OPTION>或<OPTION SELECTED>标记为弹出式列表框创建一系列表项（entries），<OPTION SELECTED>标记指示出当用户最初看到这一页时你想选择哪一个选项。

最后的窗体专用标记允许创建一个文本区。这允许你在一个小区中显示大量文本，或为用户的注释及评论提供一个大显示区。文本区使用<TEXTAREA>

和< /TEXTAREA>标记对，也能够为<TEXTAREA>标记添加一个可选的属性NAME，以便于后面恢复其内容。

现在你已经基本了解了这些标记是如何工作了。让我们来看看实际代码。程序列表 8.5 列出了一个窗体的代码示例（将产生的文件命名为 FORM.HTM 以使程序列表 8.1 中的链接正确工作）。它对能够使用的每种类型的基本控件都给出了例子，还有一个在本节末尾要更详细讨论的扩充控件。图 8.7 则显示的是从 Internet Explorer 上看到的这一页的外观，如果使用不同的浏览器，你的视图也许会稍有不同。注意，在页底部的三个按钮，使用了<CENTER>标记进行居中处理。

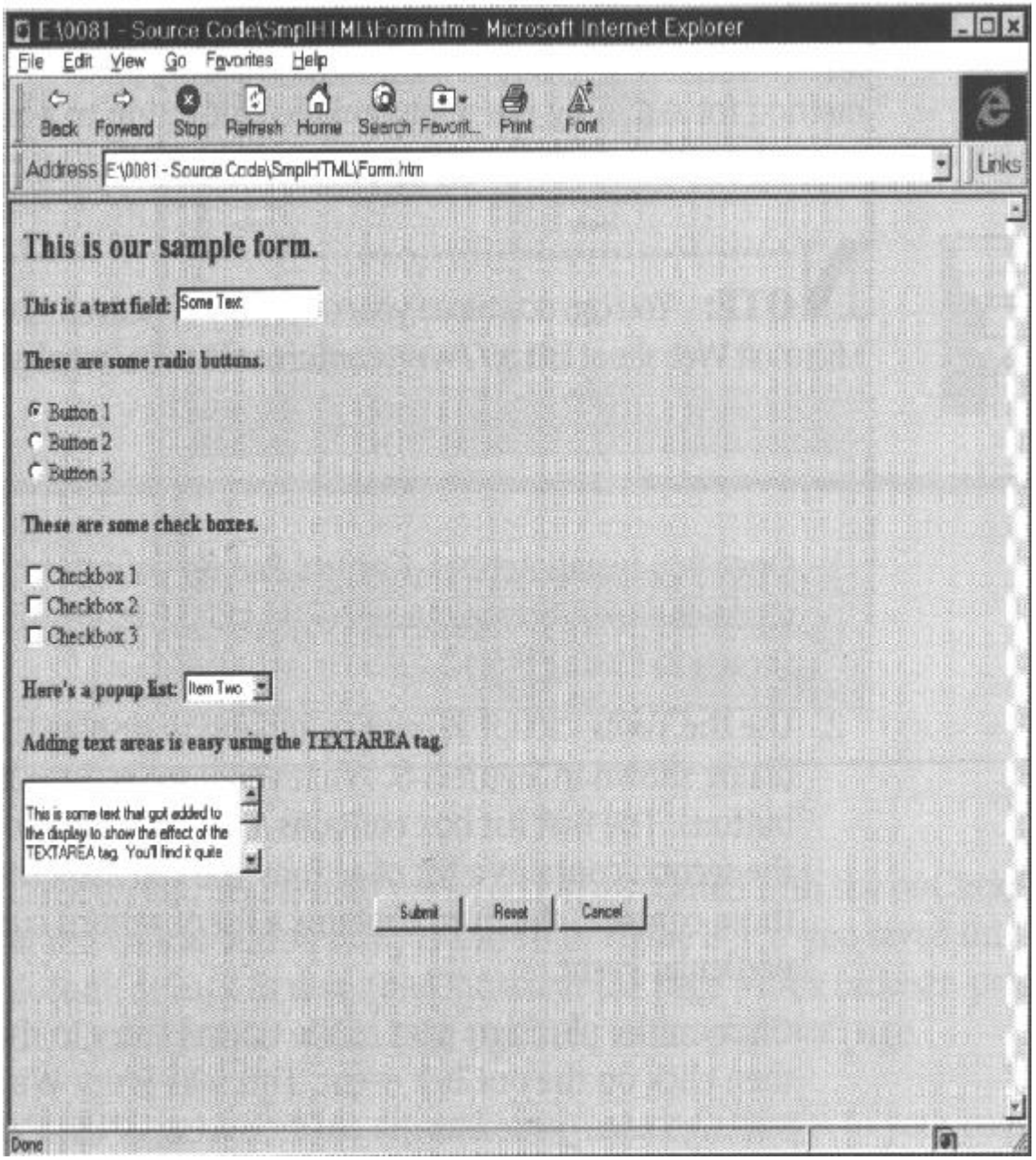


图 8.7　窗体使你从浏览者那里得到反馈

程序列表 8.5

```
<!- 这是一个非常简单的但非功能性窗体，它不是说明怎样写出通常 ->
<!- 需要的 CGI 脚本使窗体可以使用，为了看到完整的窗体的实现，->
<!- 请看类似于 http://www-msn.lycos.com 的 Web 站点->

<HTML>
<HEAD>
<TITLE>New Page<TITLE>
</HEAD>
<BODY>
<H2>This is our sample form.</H2>

<!- 使用标准控件，创建一个简单的窗体 ->
<FORM METHOD=GET ACTION="/cgi-bin/query">

<!- 创建一个文本字段 ->
<STRONG>This is a text field:</STRONG>
<INPUT TYPE="text" VALUE="Some Text" NAME="Text1" SIZE=20><P>

<!- 添加一些单选按钮。->
<STRONG>These are some radio buttons.</STRONG><P>
```

<INPUT TYPE="radio" NAME="Radio1" VALUE="Button 1" CHECKED>Button 1

<INPUT TYPE="radio" NAME="Radio1" VALUE="Button 2">Button 2

<INPUT TYPE="radio" NAME="Radio1" VALUE="Button 3">Button 3<P>

<!- **添加一些复选框。** ->

These are some check boxes.<P>

<INPUT TYPE="checkbox" NAME="Checkbox 1" VALUE="Check 1" >Checkbox 1

<INPUT TYPE="checkbox" NAME="Checkbox 2" VALUE="Check 2" >Checkbox 2

<INPUT TYPE="checkbox" NAME="Checkbox 3" VALUE="Check 3" >Checkbox 3<P>

<!- **弹出式列表和菜单需要不同的格式** ->

Here's a popup list:

<SELECT NAME="Popup 1">

<OPTION>

Item One

<OPTION SELECTED>

Item Two

<OPTION>

Item Three

</SELECT>

<P>

<!- **还能够在屏幕上创建文本区** ->

Adding text areas is easy using the TEXTAREA tag.<P>

<TEXTAREA ROWS=3 NAME="Text Area 1">

This is some text that got added to the display to show the effect of the TEXTAREA tag. You'll find it quite handy as you create Web pages with a lot of information to convey. It's very important to provide the users of your site with complete information, and this tag helps you do it.</TEXTAREA><P><P>

<!- **每个窗体都需要一个提交按钮向服务器发送数据** ->

<CENTER>

<INPUT TYPE=submit VALUE="Submit">

<!- **这一个可选的复位按钮能节省用户的时间** ->

<INPUT TYPE=reset VALUE="Reset">

<!-**这是一个标准的按钮类型，我们用它来取消窗体** ->

<INPUT TYPE=button VALUE="Cancel">

</CENTER>

</FORM>

</BODY>

</HTML>

如你所见，显示的大多数控件使用了<INPUT>标记。控件类型间的主要差别由 TYPE 属性定义，若想知道它们如何工作，可以选择任一控件。不能正确工作的唯一按钮是提交（Submit）按钮，这是因为我们还没有写出 CGI 脚本来处理窗体的信息。取消按钮还不能用，但过一会儿我们就定义它。

在源代码中还应注意到两个独特的按钮类型：提交（Submit）与复位（Reset ）。提交按钮总是执行<FORM>标记中 ACTION 属性指出的动作。若打算用标准方法处理窗体中的信息，就必须提供一个提交按钮。复位按钮总是将窗体中的全部控件恢复成初始状态，这使得用户可以从头开始重新启动窗体，而不必先离开该站点再重新进入该站点。

这个例子中的取消按钮还不能用，可以单击它，但什么也不发生。取消按钮代表了 HTML 文档的标准按钮类型，为使其工作，不得不添加一个调用某个脚本或执行某个其它动作的 ONCLICK 参数。一旦创建了这个按钮本身，ActiveX Control Pad 就提供一种容易的办法让你创建这一脚本。下列程序说明了为一个按钮（或其它控件）指定几乎任何缺省动作的快速方法。

注释 可以从 http://www.microsoft.com/workshop/author/cpad/ 的 Microsoft Web 站点为自己下载一个 ActiveX control Pad。

1. 一旦在你的机器上安装了 ActiveX Control Pad，使用 File（文件）|Open （打

开）命令打开 FORM.HTM 文件（该文件代码如程序列表 8.5 所示）。

2. 使用 Tools（工具）|Script Wizard（脚本向导）命令显示脚本向导对话框，如图 8.8 所示。看到的是三个列表框及一些按钮。第一个列表框包含了你定义的控件；第二个则包含了你可以与这些控件的事件相联系的动作；第三个包括的是当前赋给一个特定事件的动作。



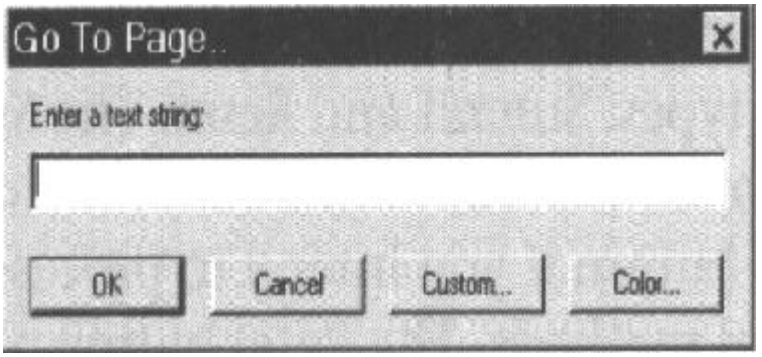
图 8.8　脚本向导显示出你的控件列表和与控件相联系的动作

3. 单击第一个列表框中 Cancel（取消）元素左边的加号，再单击 onClick 事件，

这就是让脚本向导知道你想为 Cancel 按钮的 onClick 事件 （单击事件）定义某个动作了。

4. 单击第二个列表框中的 Go To Page（到.....页）项，这表示当你单击窗体中的 Cancel（取消）按钮时，你将到另一页去。这时，你将到我们例中 Web 站点的前一页去。

5. 单击 Insert Action（插入动作）按钮，则看到如下所示的 Go To Page 对话框，在这里可以告诉脚本向导你想到哪一页去。



6. 在 Enter a Text String（键入字符串）域中键入你的 Web 站点某页名称，本例中是 SMPL_TAG.HTM。请注意不必有双引号或单引号——脚本向导自动地处理这些事。

7. 单击 OK，就可以看到 Cancel 按钮的 onClick 事件与一个动作联系起来，如图 8.9 所示。将这一文档存盘，然后用浏览器查看该页，单击 Cancel 按钮，将回到前一页去。

脚本向导自动地对你的代码作了所需的变化，在这时，它向程序列表 8.5 中我们定义的取消按钮添加了一个 ONCLICK 属性，代码如下：

```
<!- 这是标准的按钮类型，我们将用它取消窗体 ->
```

```
<INPUT LANGUAGE="VBScript" TYPE=button VALUE="Cancel"
        ONCLICK="Window.location.href    'SMPL_TAG.HTM"'>
```

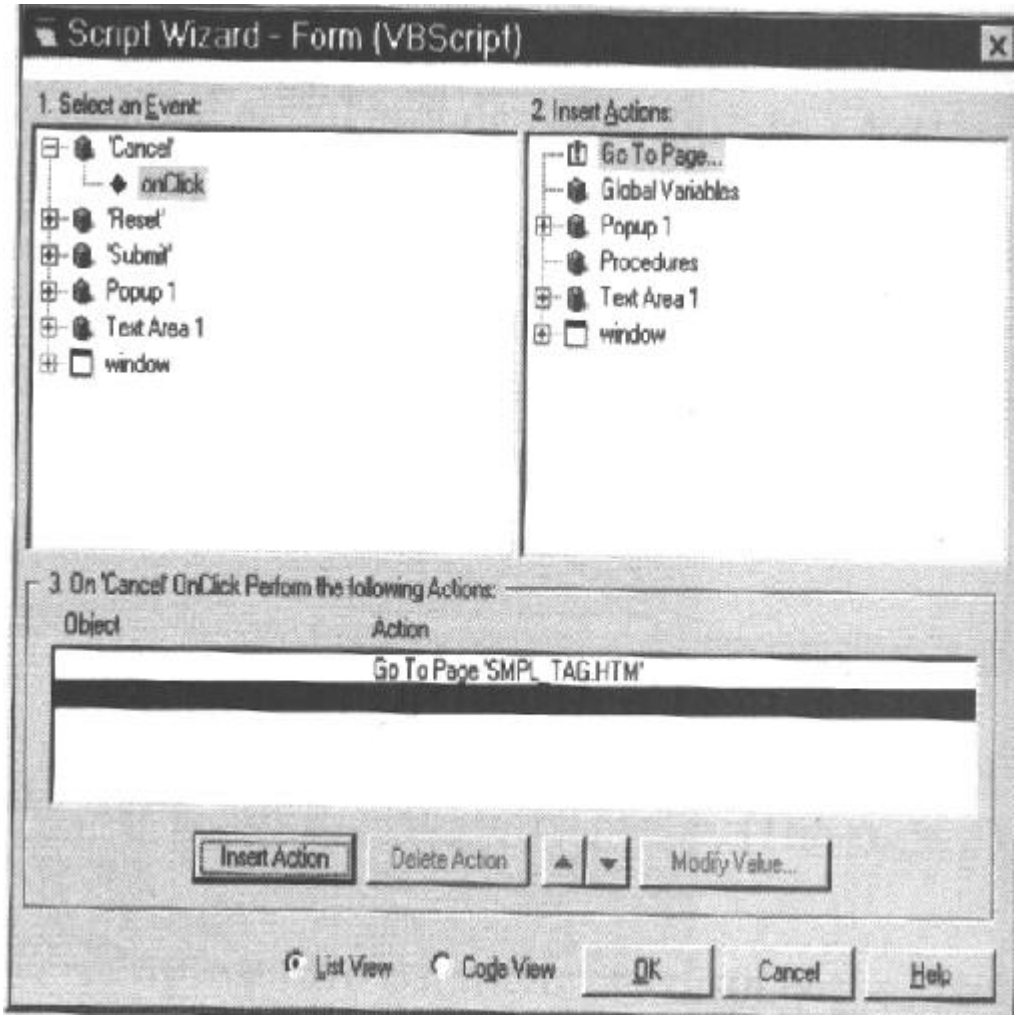


图 8.9　脚本向导通过在第三个列表框中显示事件及其相联系的动作完成你所创建的脚本

8.3 ActiveX 适用的场合

差不多整整一章用于讨论在任何浏览器上都可以使用的编码技术，这些技术对于使用 ActiveX 进行工作的 Visual C++开发人员同样重要，因为想为 Web 站点的每个方面都写出控件是不可能的。即便确实写出了所需的控件，也不可能每个人都乐意等待着将它们全部下载。到目前为止我们讨论的编码技术都是很通用的，不仅它们差不多可以在每个可用的浏览器上工作，而且它们还是与平台无关的。但 ActiveX 却不提供同一级别的支持，暂时它还是非常紧密地绑在 Internet Explorer（或带适当插件的 Netscape Navigator）上。

你可以把现在已学到的知识作为使用 ActiveX 控件进行工作的基础。和前面的几节中添加其它的控件一样，向 HTML 文档添加一个 ActiveX 控件，也需要一个标记，这个标记就是<OBJECT>。让我们来看一个基本的<OBJECT>标记。程序列表 8.6 中含有一个用 ActiveX Control Pad 创建的<OBJECT>标记的典型例子，但你可以同样容易地手工创建它。这个标记设置成使用第 10 章（“以 MFC 为基础的一个基本按钮编程例子”一节）中创建的控件进行工作，而且还需要改变 CODEBASE 特性，使之指向你的 Web 站点。我们将在以后的段落中更多地讨论 ActiveX 特性。

程序列表 8.6

```
<HTML>
<HEAD>
```

<TITLE>This Page Contains an ActiveX Control </TITLE>

</HEAD>

<BODY>

<OBJECT ID="OCXEXMPL1" WIDTH=75 HEIGHT=25

CLASSID="CLSID:D8D77E03-712A-11CF-8C70-00006E3127B7"

CODEBASE="http://AUX/CONTROLS/OCXEXMPL.OCX">

<PARAM NAME ="_Version" VALUE="65536">

<PARAM NAME ="_ExtentX" VALUE="1976">

<PARAM NAME ="_ExtentY" VALUE="670">

<PARAM NAME ="_StockProps" VALUE="70">

<PARAM NAME ="_Caption" VALUE="On">

<PARAM NAME ="_OnOff" VALUE="-1">

<PARAM NAME ="_ModalResult" VALUE="8">

If you see this message without a corresponding pushbutton, then your browser doesn't support ActiveX conponents. **（如果看到这条消息而没有对应的按钮，那么，你的浏览器不支持** ActiveX **组件。）**

</OBJECT>

</BODY>

</HTML>

注 和其它的复杂标记一样，<OBJECT>是定义的开始，而</OBJECT>是定义的结束。

先看<OBJECT>标记的第一行。ID 属性指出我们正在处理哪一种控件。我们注意在这里并没写出扩展名，Internet Explorer 缺省扩展名是 OCX。Internet Explorer 目前支持三种扩展名：OCX，CAB 和 INF。我们在本节后边将讨论各种类型的优缺点。还可以指定一个 URL 做为 ID 属性串的一部分。请注意，第一行中还包含着 WIDTH 和 HEIGHT 属性，用它们来定义控件的大小。

注 <OBJECT>标记能识别 OCX、CAB 和 INF 这三种文件扩展名。

和本地机器中每个 ActiveX 控件一样，从 Internet 下载的 ActiveX 控件也具备 CLASSID 属性，这是 Windows 识别一个特定控件的唯一编号，该编号存贮在 Windows 注册表中。在后面将会看到，如果你使用的控件在用户的机器中已经安装，这时，这一特色就能帮助减少控件的下载时间。

添加 CODEBASE 属性的作用是，如果主机上还没安装该控件的话，就指示到你的因特网站点的某处去找到这个控件。如同已在 AUTOEXEC.BAT 已放入的远程 PATH 语句那样，把它也想像成一个远程 PATH。如果不向<OBJECT>标记添加 CODEBASE 属性，那么，就不能有效地使别人从你的 Internet 站点下载该控件。下一节我们会看到这个特色是如何工作的。

现在我们已定义一个对象，一大串<PARAM>标记紧随其后。这些标记定义了在向用户显示该控件时如何配置它。<PARAM>标记总是包括两个属性：NAME 和 VALUE。NAME 特性定义了你想设置的参数名称，而 VALUE 属性则向参数赋一个值。所有值都用双引号括起来，既便是数值也不例外。

程序列表中最后三个参数，是任何环境下使用该控件时能够设置的永久性特性。Caption（标题）特性改变按钮顶部的文本，就好象你是在 Visual C++ 中使用该控件一样， OnOff 特性是这个控件所特有的——它允许你创建一个 On/Off 按钮，第 11 章中我们要创建一个实际控件，到那时就会看到它如何工作。ModalResult 参数允许控件返回一个基于其当前标题的值，例如，按钮标题是 On 时，控件返回为 8。

还有一些通常你不无须关心的参数，但把控件作为 Web 页一部分时却需要进行处理。_Version 参数在下载过程中起作用，它帮助浏览器对照服务器上的控件版本来确定客户机的控件版本是否已经过时。_EntentX 和_EntentY 特性则在 Web 页上将控件定位，最后，_StockProps 属性定义控件的族系特性——通常不需设置它们。

接下来你将看到的是一个串。支持 ActiveX 的浏览器将会忽略这个串。只有当客户机不支持 ActiveX 时，这个串才被显示出来。实质上它从另一方面告诉用户，如果它们的浏览器提供了正确的支持，那么，它们立即就能看到一个 ActiveX 控件。你不必受限于一个串，尽管你总是可以在这里放上一个等价的 HTML 控件。例如，这个 ActiveX 控件是一个能用 HTML 代码代替的按钮，尽管它不再这样灵活了。

前面我们谈到可以从 Internet 上下载的文件有三种类型：OCX、CAB 和 INF。OCX 文件格式，又称为可移植的可执行文件（PE）格式，使你能用最终形式向客户机发送控件。在另一边不需附加处理，从长远看来，意味着你遇到的支持问题会更少。CAB 文件格式与 Microsoft 发行(shipping)产品(如 Windows)时使

用一样的格式。它独到的优点是让你在一个单一包中发行多个控件。这种格式还提供了文件压缩功能，从而减少用户的下载时间。缺点是太复杂，你不得不创建几个安装文件，还包括一个 INF 文件。另外，一旦下载完成之后，用户需要等待 CAB 文件解压缩，然后安装它们，这就使得这种方法更可能会使用户忍受不了，并在它们试图取消这种处理过程时引起许多问题。INF 格式允许选择性地安装一个或多个控件，它使用的 INF 文件的格式类似于 Windows。用户访问你的站点时，就会看到某种类型的安装屏幕显示。它的缺点（downside）是仍然要创建 INF 文件。使用 INF 文件，意味着当用户等待页下载时，让他感到完全厌烦的机会会更少，而且他们至少是过程的一部分。我们将在本章后面讨论 INF 格式（参见“使用 DIANTZ.EXE 建立组件下载(CAB)文件”）。

8.4 COM+：Internet 的未来

对 Windows 进行的任何技术讨论，必然要包括对组件对象模型 COM 的讨论。COM 是一种对象技术，已成为 Windows 的全部（技术）的基础，并且还是 Microsoft 向 Internet 进军的一部分。COM 不仅与用户相互作用，而且还是创建类似于应用程序中使用的按钮之类的组件的技术基础。换言之，不以某种方式讨论一下 COM，就不能开始对 Internet 的 Windows 或 Microsoft 版本进行讨论。

就在一年前，ActiveX 还是表达 Microsoft 进军 Internet 的新版 COM 的中心词汇。但是，现在已经证明，ActiveX 太庞大（尽管 ActiveX 控件远远小于被它

取代的 OCX），太依赖于浏览器，而且对大多数人来说又有太多的安全风险。另外，Microsoft 市场部正在用 ActiveX 来描述 Microsoft 正在发布的几乎每一项新技术，这样只能使客户更加迷惑。

当 COM+及分布式网际网应用程序（Distributed interNet Application，DNA）出现在 Internet 时，就又有了两个新鲜而又激动人心的术语让你学习了。ActiveX 不再是描述 Microsoft 的 COM 技术诸个方面的术语，它仅关系到在应用程序或内部网站点中创建并使用的组件。如果你正在谈论使能技术或 DNA 技术，如果你正在谈论 COM+技术的进展情况，那么在 Internet 上干任何事现在都称为 COM+。

WEB 链接 可以在 http://www.microsoft.com/com/default.htm 上得到 Microsoft 种种最新的 COM 技术。这个站点可使你一直跟得上 Microsoft 正在引入的各种与 COM 相关的技术发展，以及预期它们是如何发展的。

事实上，作为 C++程序员，需要了解相当多的基于 COM 的技术。下列章节将快速地浏览一下这些技术，并讨论如何将它们放入一般方案中。显然，在 Microsoft 升级其 COM 技术时，这种讨论也会有所变化。

分布式 COM（DCOM）——网络接口 COM 的这一版允许组件和应用程序在网络上进行通信。DCOM 设计得可以在相当多的网络协议——包括 Internet 使用的 TCP/IP——上进行通信。DCOM 为开发人员所能做到的，是允许客户的应用程序只要使用标准的 COM 接口，就可和运行于服务器上的服务直接通信。实际上，这就意味着你可以设计一个控件或一小段程序，

使得客户机的应用程序从服务器接收数据，然后在本地对这些数据进行计算。另外，DCOM 使得我们可以在客户机和服务器间进行安全通信，而不需关心所用的协议。

COM+——功能　标准 COM 接口扩充。它是现有的 COM 标准的扩充，使开发人员能利用特殊的 COM+调用来设计高级的控件。COM+提供的最重要的 COM 扩充是数据捆绑，允许组件提供数据库访问。添加数据库访问，意味着可以使用 COM+为能访问远程数据的 Internet 建立组件。COM+还是分布式网际网（DNA）体系结构的基础，而 DNA 是 Microsoft 的最新的 Internet 版本。

理论上，DNA 允许 COM 组件可以在任意平台上执行，并且允许开发人员可以使用任意语言写出该组件。可以使用任意语言和任意平台进行工作的这部分能力，来源于一个称之为侦听器的 COM+新特色。使用侦听器允许 COM+ 组件可以在运行期间调用多种服务，而不是依赖于某一具体服务。你将使用侦听器接收和处理与实例创建、调用与返回、出错以及实例删除有关的事件。侦听器还提供了处理事务及系统监督的机制。

Microsoft 事务服务器（MTS）——功能　COM 服务器。信不信由你，MTS 是 COM 的一部分，它允许开发人员创建“轻型(lite)”控件，这种控件集中于处理组件背后的事务逻辑，而不是处理组件所需要的全部接口。建立轻型控件意味着能够更快地创建它们，使用时下载时间更少。MTS 将成为 IIS4.x 的一部分。

ActiveX——功能　组件建立。ActiveX 最初是作为 Internet 的一项新技术而

引入的，它是一种组件建立技术，它是 Microsoft 组件技术的第三个版本。将 ActiveX 从以前的 OLE 控件扩充（OCX）中分离出来的原因之一是，可以在类似于 Internet 的高级潜在网络中使用它们。它们还设计得可以与浏览器一起工作，尽管在写出本书时真正使用它们的唯一浏览器只有 Internel Explorer。ActiveX 控件的特色还包括增量绘制和代码签名，使得用户可以在控件执行前识别出它们的作者。

8.5 从 Internet 上下载 ActiveX

本章中介绍的大部分 HTML 标记的作用是很明显的，浏览器看到标记后就把标记的要求显示出来。整个过程简单易懂。但是 ActiveX <OBJECT>标记却相当不简单。当浏览器看到一个<OBJECT> 标记后会发生什么事? 我们知道它不会只是简单地在屏幕绘制控件——ActiveX 控件提供了比这多得多的功能。

Microsoft 已作出了一个新的 Windows API 调用供浏览器使用——CoGetClassObjectFromURL。当浏览器看到一个 < OBJECT> 标记时，它先分析出 CLASSID 、CODEBASE 和_Version 参数，并把这些参数传送给 CoGetClassObjectFromURL。如有必要，这个 API 可以来完成控件的下载、验证和安装。这个 API 调用做的第一步，是检查当前注册表中是否包含着对 CLASSID 的引用。你可以在注册表的 HKEY_CLASSES_ROOT|CLSID 关键字中找到这一引用。如图 8.10 所示。

WEB 链接 关于<OBJECT>标记的规范及其相联系的 API 调用处于经常变动之中，若想得到有关<OBJECT>标记的最新信息，请参看http://www.w3.org/TR/WD-object.html。

如果 CoGetClassObjectFromURL 找到了安装在客户上的 ActiveX 控件的实例，它就检查其版本号。当安装在客户机上的控件版本号大于等于 HTML 页上指出的版本号时，CoGetClassObjectFromURL 就装载本地控件，而不再从 Internet 上下载这个控件。控件装载完后，CoGetClassObjectFromURL 就为它创建一个类工厂（Class Factory），并把类工厂传送回浏览器（一个类工厂和其它工厂一样工作，它产生出一些项，在上述情况下，它允许浏览器产生对象的一个实例）。否则，API 调用就请求浏览器从 Internet 站点下载代码。

下载是一个异步处理过程——Windows 等待下载完成的过程中能完成其它与 ActiveX 有关的任务。一旦浏览器完成下载并在必要时解压缩后，就调用 Windows 信任提供者服务函数 WinVerifyTrust。这个服务到 ActiveX 控件内部检查并确定是否有签名块。签名块包含有作者姓名、公开密钥以及控件内容加密摘要等。可以把 ActiveX 控件签名块看成是支票的签名、驾驶执照或一份合约，它不仅指出你是谁，而且还验证你是不是你说的你是谁(you are who you say you are)。若 WinVerifyTrust 调用找到了一个签名块（也称为证书），它就对证书进行证实。每个证书能够包含一个父证书的名称。WinVerifyTrust 沿着证书的分层树上溯直至到达根证书。然后它就在可信任的根证书列表中找这份根证书。若能找到，CoGetClassObjectFromURL 自动地装入控件并为它创建一个类工厂。否则，用户就会收到一条该控件不可信的消息（当我们在第 10 章中创建

一个 ActiveX 控件并在 Web 页中使用时你会看到这一消息的）。

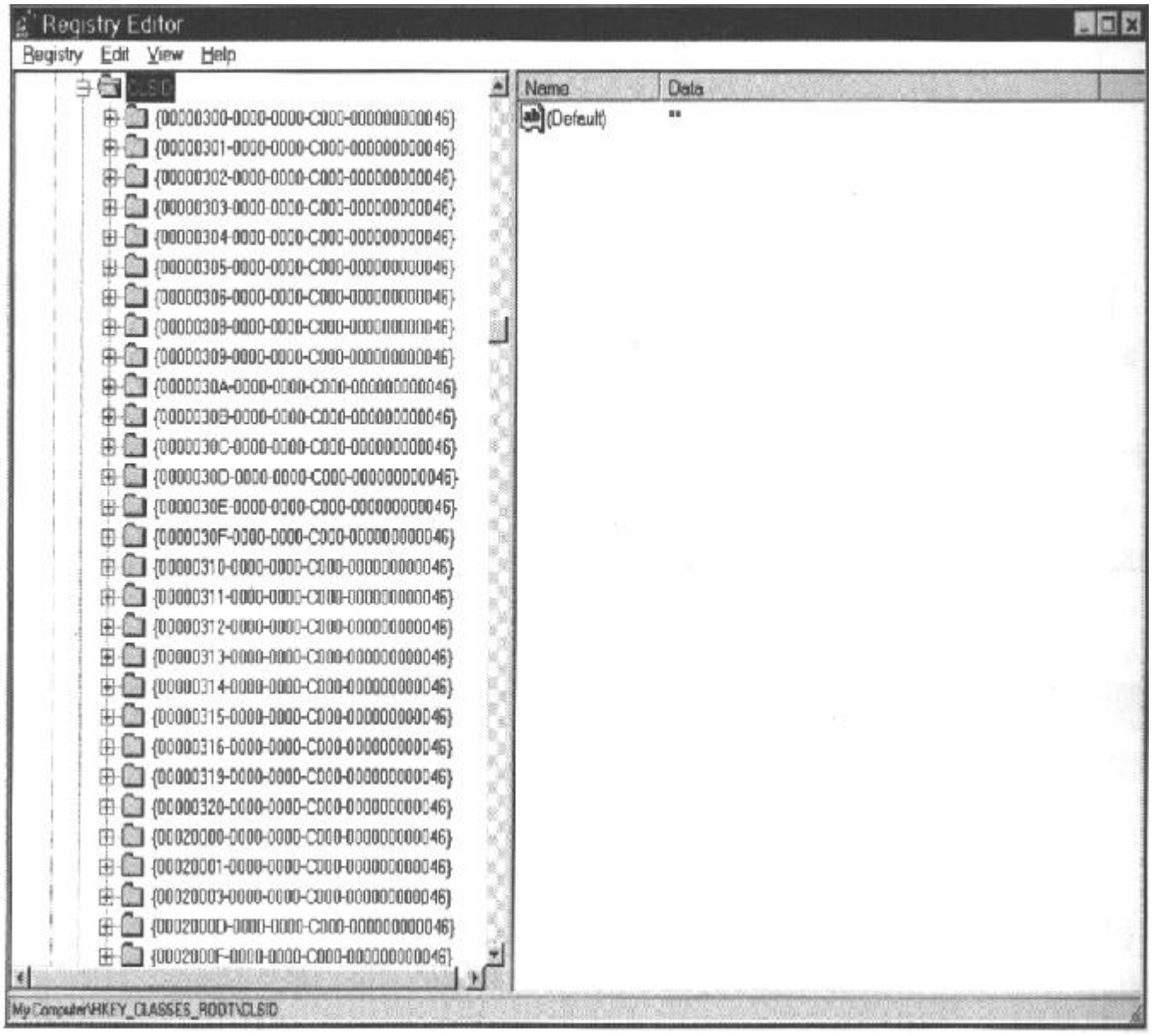


图 8.10　Windows 在 HKEY_CLASSES_ROOT|CLSID 关键字中存放 ActiveX 控件的类标识符

注　公开密钥的作用如同共用房屋或公寓的安全门钥匙——它使你能进入

大楼。私有密钥则如同公用房屋或公寓的室门钥匙——它使你进入你的起居室。

即使一个 ActiveX 控件不出现在可信任的列表中，用户仍可选择是否安装这个 ActiveX 控件。若选择安装这个控件，用户就会被问及是否想把控件作者添加到他们的信任列表中去。向列表添加作者姓名，意味着同一作者的任一新控件将会立即得到承认。我们将在第十四章中更详细地讨论安全问题。

这些控件安装在什么地方呢? 理论上，控件设计者可以把控件放在任何位置。但是，大部分从 Internet 下载的 ActiveX 控件，都不会自动地出现在用户的类似 System （系统）文件夹这样的地方。实际上，它们被添加到名为 OCCACHE 的一个特殊文件夹中，这个文件夹可以出现在用户机器的任何位置。通常你可以在 Windows 主文件夹、/SYSTEM 文件夹或用户的 Internet 文件夹中找到这个文件夹。（一些浏览器还使用 ActiveX Control Cache 文件夹，它位于存放 ActiveX 控件的 Window 主文件夹中）。

注　可以找到从 Internet 上下载的 ActiveX 控件的最普遍的位置是 OCCAHE 文件夹。

装入控件还不是工作的全部，控件在安装过程中还要对自身进行注册。在大多数情况下，这意味着要调用 DllRegisterServer API 函数。完成安装及注册后，CoGetClassObjectFromURL 函数把与控件相联系的类工厂传回浏览器。浏览器使用类工厂创建一个对象实例，并对<OBJECT>标记传送的参数进行初始化，（必要的话）将控件显示到屏幕上。

ActiveX 与 Netscape Navigator

如果你管理着一个内部网站点，你总是能控制让谁访问站点以及它们应该用什么浏览器来进行访问。这时，如果想使用 ActiveX，那么，你就只需要简单地要求每个人都使用 Internet Explorer 就可以了。但是，对于 Internet 站点来说却做不到这一点。所以就不得不确保站点支持尽可能多的浏览器。Internet Explorer 直接支持 ActiveX，而 Netscape Navigator 却不是这样。幸运地是，有一种可供选择的办法使 ActiveX 可以在这样的环境中工作。

NCompass Labs,Inc.公司生产了一种称为 ScriptActive 的插件，它允许使用 Netscape Navigator 3.0 看到含有 ActiveX 控件的 HTML 页。在写本书时，NCompass 的插件和 Internet Explorer 3.0 的 ActiveX 控件在能力上是类似的。在使用这个插件之前，可能需要花些时间研究一下它。NCompass 在 http: //www.ncompasslabs.com/为 ScriptActive（及其它产品）提供支持。

不幸的是，使用当前版本的 ScriptActive 不太简单。你不得不操作<OBJECT>标记的一些特殊格式来让它工作。还算幸运，这个插件中还包含有可以完成大部分工作的 HTML 转换实用程序，名称是 NConvert。程序列表 8.7 显示的是使用这个实用程序创建标准的 ActiveMovie 控件所产生的结果。

程序列表 8.7

```
<!- 插入一个 ActiveMovie 控件 ->
<OBJECT ID="ActiveMovie1" WIDTH=267 HEIGHT=73
        CLASSID="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A"
```

```
        CODEBASE="http://aux/controls">
        <PARAM NAME="_ExtentX" VALUE="7038">
        <PARAM NAME="_ExtentY" VALUE="1931">
<EMBED NAME="ActiveMovie1" WIDTH=267 HEIGHT=73
        CLASSID="CLSID:05589FA1-C356-11CE-BF01-00AA0055595A"
        CODEBASE="http://aux/controls"
        TYPE="application/oleobject"
       PARAM__ExtentX="7038"
       PARAM__ExtentY="1931"
></OBJECT>
```

警告 一旦从一个特定的<OBJECT>标记向一个<OBJECT>标记和<EMBED> 标记组合的转换，就不要通过单击 HTML 页上邻接元素的<OBJECT>标记来编辑 ActiveX 控件。在某些情况下，编辑器会改变<EMBED>标记，使之不能正常工作。最好创建自己的Web 页，并在 Internet Explorer 中测试它，然后添加所需的<EMBED>标记，使它通过 NCompass ScriptActive 插件来进行工作。

请注意，NConvert 采纳了<OBJECT> 标记的信息并在同一信息上添加一个< EMBED> 标记。Navigator 能识别出 <EMBED>标记是要向插件中传达某种信息，插件用 ScriptActive 完成它的工作。因为 Internet Explorer 不能理解上下

文中的<EMBED>标记，多出来的这些信息不引起任何问题。这一部分额外的代码的（存在）理由是相当明显的。ScriptActive 插件需要一个特殊的 TYPE 属性，使用它指定对象的因特网 MIME 类型。根据供货商的说法，ScriptActive 将来的版本中不再需要这一附加信息，并且能够不加修改地使用<OBJECT>标记。

技巧 一旦 NCompass ScriptActive 插件安装成功，用右键单击文件时，就会看到 NConvert 关联菜单项，这个项包含两个选项：NConvert（转换）和 Set Destination Folder(目的文件夹设置)。使用 Set Destination Folder 这个选项，可以告诉实用程序，你想把转换过的文件放于何处，如果不想用转换过的文件复盖原来的文件，那么，就要指定一个特殊目录。Convert 选项的作用是插入产品的当前版本所需的所有<EMBED>标记。

ScriptActive 还支持 VBScript，但是它是通过添加一个外部的 AXS 文件来实现的。一定要保证 AXS 文件与包含 VBSCript 的 HTML 文件在同一个目录中。NConvert 添加一些 JavaScript 行，从而建立 Navigator 和 ScriptActive 之间的联系。ScriptActive 收到请求后，就在 AXS 文件中寻找合适的 VBScript 代码。不幸地是，ScriptActive 对 VBscript 的支持相当有限，在使 AXS 文件工作之前，请对这个文件中的 VBScript 代码进行再加工。如果计划在一个站点上同时使用 Internet Explorer 和 Navigator，那么，最好的方法是优先使用 JavaScript 而尽量避免使用 VBScript。

使用 ScriptActive 带来的另一个问题是，它不能总是让你访问与对象相联系的属性。事实上，支持的级别甚至会随机器而变，具有相同配置且使用相同硬

件的两台机器，在一个具体的 Web 页上，也可能显示出不同的结果。不同的控件看起来也提供可变化的访问级别，复杂的控件往往比简单的控件出的问题更多。考虑到 Internet 处于不断变化之中且 ActiveX 技术相对比较年轻，出现这些问题也不值得大惊小怪。你能够期盼获得的最好结果是显示控件于屏幕，至于它能不能实际工作则是另一回事了。

尽管存在这些问题，使用 ActiveX 控件的前途是太光明了，不允许忽视它。类似于 ScriptActive 的第三方厂商的插件，差不多是在 Web 站点上能够使用 ActiveX 的唯一途径，并且仍能保证访问站点的大多数人，能够实实在在地看到你所提供的内容。有传言说，Netscape Navigator 的下一版将直接支持 ActiveX，但在今天，ScriptActive 依然是你最好的解决方案。

8.6 增强 HTML 的功能

我们可以对一个 Web 站点进行许多方面的增强(Enhancement)。实际上，你将在 Internet 上看到的各种增强，大概能塞满整整这一本书。对 Web 页使用增强，会产生两方面的问题：第一，大部分技术在数据格式上需要一些约定，你不得不考虑，一个访问者是否真想下载数据，从而产生所要的效果；第二，可以发现，一些增强在这个浏览器工作得不错，而在另一个浏览器却不行。在后面的章节中我们将陷入这些不兼容性之中，所以从现在开始，我不再对这一点进行反复说明。

对 Web 站点能够进行的最流行的两种增强是声音和动画。到目前为止，声

音在二者之中更为流行。但这里我们并不是要看完整的(full-fledge)CD 或优雅的传真图像。声音以声音字节流的形式出现，声音字节流是一小段声音，可以添加到你试图在 Web 站点上获得的效果中去。大多数情况下，人们使用 WAV 文件向站点添加声音，也在一些人使用 MIDI 文件添加音乐。不论你使用声音支持的哪一种形式，当决定添加多少声音时要小心谨慎，少一些为好。应该给人关闭声音的选择，以便于它们在办公室工作时能访问你的站点。

使用字幕是向 Web 页添加动画形式的一种途径。字幕所做的一切就是显示一个滚动的消息或图形图像。这一标记的文本处理能力受到一些限制，任何添加到<MARQUEE>标记中的“花体”(fancy)文本将不在屏幕上显示，你开始使用的文本就是一直不变的文本（字体）。这就意味着，在实际显示字幕之前，要进行你想做的格式化（这也正是许多程序员宁愿使用图像的原因）。Microsoft 还提供了字幕的 ActiveX 版本，作为 ActiveX Control Pab 实用程序的一部分。

警告 使用<MARQUEE>标记时总是要使用 LOOP 属性，因为如果不使用 LOOP 属性，支持<MARQUEE>的有些浏览器就会做出奇怪的反应。你第一次使用这个特定标记时，注意收集用户反馈是非常重要的，因为在旧式的浏览器上，它非常容易引起麻烦。虽然有些不完美，但进行一些小的改动，也许就能使这个标记在你的站点上确实支持这个标记的每个浏览器工作。

在不用增加太多开销（但确实要增加一些）的前提下，动画确实能为站点增光。问题当然是要花费时间来创建某种动画例程。一些人也许试图写一个脚本或创建一个 ActiveX 控件来完成这一工作，但这可用更好的方式来解决。如

果你不想创建花哨的动画方案的话，就不必被迫去这样做。只需使用一个动画 GIF 即可。如果你和我一样，也许会问动画 GIF 到底是什么呢? 因为大多数程序员从未见到过它。实际上，如果你在 Internet 上花费了很多时间，也许已经看到了一两个这样的文件。一个动画 GIF 是由一组图像组成的，这些图像在你的 Web 站点上一次显示一幅。如果非常细心，你就能够使用动画 GIF 创建动画效果，既增加了快乐，又收藏了资源。最好的消息是绝大多数浏览器都完全支持动画 GIF。

WEB 链接 在许多 Internet 站点可以看到活动的动画 GIF 。一个更有趣的站点是 http://www.wanderers2.com/rose/animate.html，这个站点提供了一个站点索引，访问这些站点就能看到多种动画 GIF。到各种站点去转一转，会帮助你明白什么能做，什么做不了。你还可以下载动画 GIF 向导，从而使你自己的动画 GIF 在线，并能学习有关怎样制作动画 GIF 的全部知识。

利用声音和其它效果使 Web 页面更引人入胜

(WAV 文件中的)声音片段(bites)对于装饰一个 Web 站点大有好处。但是，如果每次用户完成一个最小的任务时，你都连续地演奏声音，就会令用户心烦意乱。因此声音片段要短且要用在点子上。添加声音要像厨师添加调料一样，少许的调料对烧出味道鲜美的菜肴大有裨益，用多了调料反而使人倒胃口。

Web 站点的东西越小越好。最近到一些站点的访问表明：一些站点五彩缤纷、非常有趣且下载又非常之快。看了一下代码，反映了一个共同现象，那就

是所有作者都用表格和框架(frame)将页面分成更小的区，每个小区使用小图标和图形，使页面看起来非常漂亮。另外，作者还用某种谐调的绘制方案，使得每个框架(frame) 有独特的外观，并且，页面也没有显得俗气。

下面来看一个示例页面，这一页对我们在本章中已讨论过的概念作了示范性说明。程序列表 8.8 到 8.12 显示的是怎样制作这一页，注意到这里包含了一些小的图形和框架(frame)来组织信息。在这里还可以看到本章中学习过的其它技巧（也有一些新东西）。图 8.11 显示了这一页看起来是什么样——至少从灰度上，你需要适当地使用颜色来构造页面。

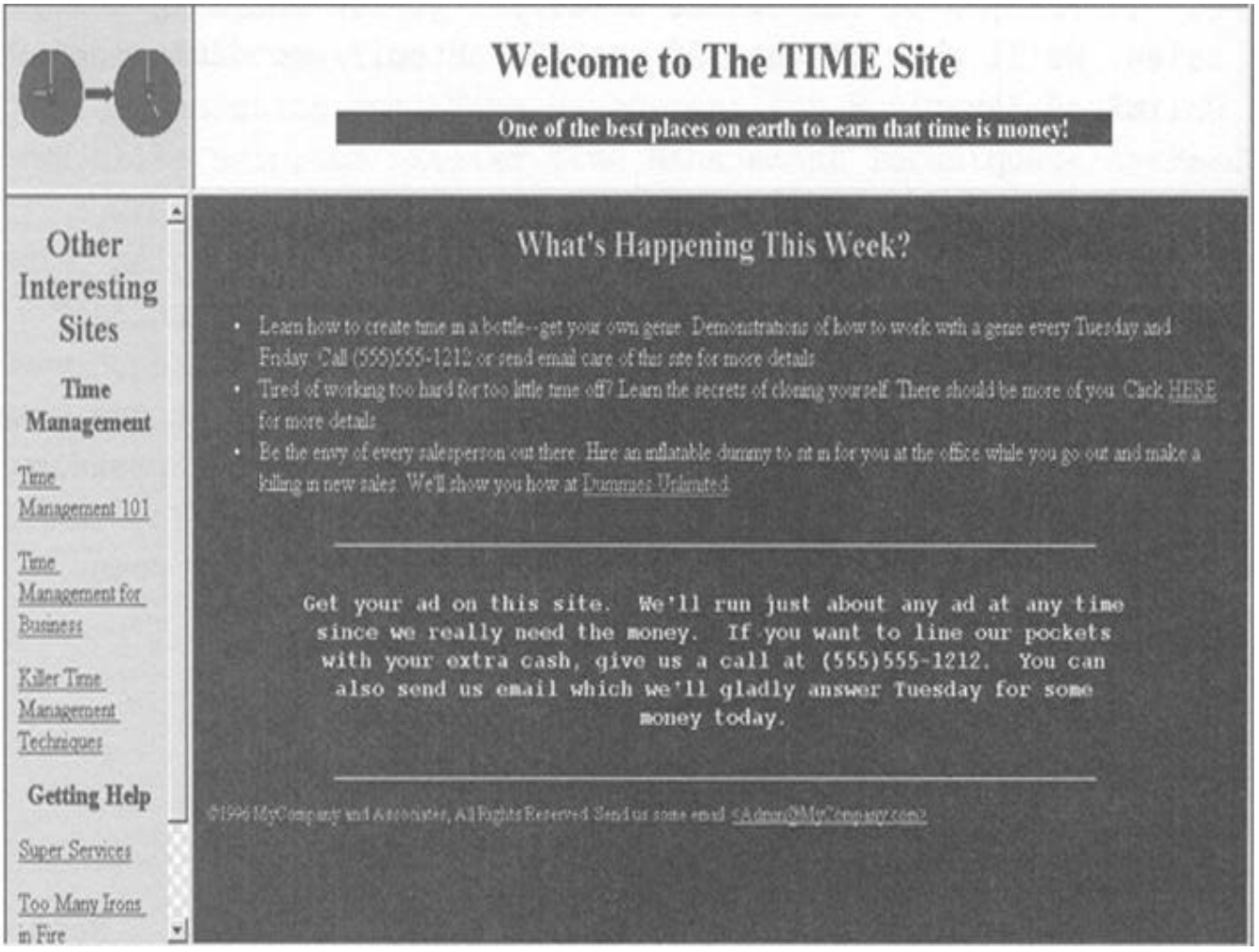


图 8.11　我们例子中的 Web 页表明：不必使用大量的资源，只要将一些简单标记组合起来就能创建一个令人感兴趣的 Web 页面

程序列表 8.8

```
<!-FRAME.HTM->
<HTML>
<HEAD>
```

<TITLE>Sample Web Page</TITLE>

</HEAD>

<!-**设置框架参数**->

<FRAMESET COLS="15%, *" ROWS="20%,80%">

<!-**加入一个图标**->

<FRAME SCROLLING=NO SRC=Icon.HTM>

<!-**创建一个标题**->

<FRAME SCROLLING=NO SRC=Heading.HTM>

<!-**列出可能会感兴趣的一些其它站点**->

<FRAME SRC=OthrSite.HTM>

<!-**显示页的主要内容**->

<FRAME SRC=Main.HTM>

</BODY>

程序列表 8.9

```
<!-HEADING.HTM->
<HTML>
<HEAD>
<TITLE>New Page</TITLE>
</HEAD>
<BODY BGCOLOR=WHITE>

<!-为我们的站点定义标题和字幕->
<CENTER><H1>Welcome to The TIME Site</H1>
<H3><FONT COLOR=WHITE>
<MARQUEE BGCOLOR=BLUE BEHAVIOR=ALTERNATE WIDTH="75%"
LOOP=INFINITE>
  One of the best places on earth to learn that time is money!
</MARQUEE>
</H3><FONT COLOR=BLACK>
</CENTER>

</BODY>
</HTML>
```

程序列表 8.10

```
<!-ICON.HTM->
<HTML>
<HEAD>
<TITLE>New Page</TITLE>
</HEAD>
<BODY BGCOLOR=YELLOW>

<!-显示我们的图标->
<IMG SRC=TimeIt.GIF ALT="From 9 to 5">


</BODY>
</HTML>
```

程序列表 8.11

```
<!-OTHRSITE.HTM->
<HTML>
<HEAD>
<TITLE>New Page</TITLE>
</HEAD>
<BODY BGCOLOR=YELLOW>
```

<!-显示标题->

<CENTER>

<H2>Other Interesting Sites</H2>

</CENTER>

<!-显示时间管理链接->

<P><P></CENTER><H3>Time Management</H3></CENTER>

Time Management 101<P>

Time Management for Business<P>

Killer Timer Management Techniques<P>

<!-显示添加的帮助链接->

<P><P><CENTER><H3>Getting Help</H3></CENTER>

Super Services<P>

Too Many Irons in Fire<P>

Business Help to Go<P>

</BODY>

</HTML>

程序列表 8.12

```
<!-MAIN.HTM->
<HTML>
<HEAD>
<TITLE>New Page</TITLE>
</HEAD>
<BODY BGCOLOR=BLUE LINK=YELLOW VLINK=SILVER>

<!-为进入站点的人演奏声音->
<!-Internet Explorer 标记>
<BGSOUND SRC="TwilZone.WAV" LOOP=1>
<!Navigator 标记>
<EMBED SRC="TwilZone.WAV" HIDDEN=TRUE LOOP=1>

<!-显示标题->
<CENTER><FONT COLOR=YELLOW>
<H2>What't Happening This Week?</H2>
</CENTER><FONT COLOR=WHITE>

<!-显示当前事件列表->
<UL>
```

Learn how to creat time in a bottle- - get your own genie.
demonstrarions of how to work with a genie every Tuesday and Friday.
Call (555)555-1212 or send email care of this site for more details.
Tired of working too hard for too little time off? Learn the
secrets of cloning yourself. There should be more of you. ClickHERE for more details.
Be the envy of every salesperson out there. Hire an inflatable
dummy to sit in for you at the office while you go out and make killing
in new sales. We'll show you how at Dummies
Unlimited

<!-**为将来的潜在销售提供一个广告**->
<CENTER><HR WIDTH=75%></CENTER>
<H3><CENTER><PRE>
Get your ad on this site. We'll run just about any ad at any time since
we really need the money. If you want to line our pockets with your
extra cash, give us a call at (555)555-1212. You can also send us email
which we'll gladly answer Tuesday for some money today.
</PRE></CENTER></H3>

```
<!-请给我们发电子邮件->
<CENTER><HR WIDTH=75%></CENTER>
<FONT COLOR=YELLOW SIZE=-1>
&#169;1996 MyCompany and Associates, All Rights Reserved.  Send us  some
email:<A HREF="mailto:admin@mycompany.com">&lt;Admin@MyCompany.com&gt;
</A>
</BODY>
</HTML>
```

注释　这个例子并没有涵盖你所能做的一切事情，下一节中我们将讨论动画 GIF。

正如你所看到的，这个站点的画面还不是很完美，但是，它大概比你过去访问过的一些地方更令人感兴趣。让我们用些时间把这些代码分析一下，看看其中包括的更具诱惑力的部分。你注意到的第一件事是站点分成了 5 个文件，可以发现，通过使用分离开的文件，使得对具有框架的站点进行维护更加容易。另外，站点上的所有页可以使用相同的标题及图标文件，这样可以为用户节约下载时间。程序列表 8.8 中的代码完成的全部工作，就是把浏览器工作区分成若干框架，然后再定义每个单元属于谁。应注意，能够明确地用百分数或*号来定义每个框架的区域。浏览器首先按百分比把可见的视屏分给定义的区域，然后再把剩余区域平均分配给其它单元。使用这一特色能够在 Web 页上强制实现某一方面的比例。

注　使用(*)号定义某框架时，其意义是浏览器为已指定的其它框架分配空

间后，该框架占用剩下的窗口空间。

程序列表 8.9 含有 Web 站点的标题。这里的代码中，唯一令人感兴趣的就是字幕区域颜色的使用。在整个站点谐调着色可以使之美观，而且又不花费用户多少下载时间。还应注意到，这个字幕在前后跳跃而不是兜着圈转。BEHAVIOR 属性就是负责这种变化方式的。使用多种属性来提供特殊效果，能够使你的站点更加令人注目。还应注意这是站点上唯一使用动画的地方，再多了就会使人反感。

可移植性 <MARQUEE>标记仅在 Internet Explorer 中工作。若使用 Navigator，仍可以看到标题区，但没有滚动的字幕。解决这一问题的较好的方案是，创建一个 Java applet 来激活<MARQUEE>标记的行为。也可以创建一个 ActiveX 控件来完成这件事，但不得不使用类似于 NCompass ScriptActive 之类的插件使控件与 Navigator 兼容。

程序列表 8.10 中，只有一个关于我们站点的图标的标记。须注意<BODY> 标记一定含有一个 BGCOLOR 属性，从而使页的背景色与图标能匹配起来。这时，若用户改变了显示区的大小，它们就将看不到图标下面的页面颜色了。

许多站点包含有到其它站点的链接，但总体上不妨碍站点的工作。程序列表 8.11 中代码的目的就在于此。它在不占用 Web 页本身空间的前提下，组织了从当前站点到因特网其它站点的链接。这样做，不仅使信息更容易找到，而且还可以把它作为你自己站点的一个分类索引。

在程序列表 8.12 中可以找到一些有趣的编码技巧。第一，标记的属性如何

才能用于全页以获得特殊效果。例如，应注意如何使用<PRE> 标记为页面中间的注释提供特殊格式。还应注意到这一页使用了反视频色（暗背景，亮文字，而不是通常的那样）这一特殊效果，使页面的主要部分从它周围的辅助区域中突出出来。Internet 的几个站点都使用了这种相同的效果。例如：

http://www.pacbell.com.

这一页还使用了<BGSOUND>这个新标记，使得用户进入你的 Web 站点时，就为他演奏一小段声音。特殊的 LOOP 属性用于定义声音片段应该演奏几遍。仅使用< BGSOUND>标记本身，只演奏一次，通常最好包含进 LOOP 属性，以防止出现不兼容的问题。Microsoft 也许会决定，今后某时开始，你必须明确地指示<BGSOUND>标记应把声音演奏几遍。

可移植性问题 请注意，<BGSOUND>标记是 Internet Explorer 专用的，必须用<EMBED>标记使 Navigator 也能演奏声音。有大量的与此相类似的标记——Internet Explorer 不认识 Navigator 标记，而 Navigator 也不认识 Internet Explorer 的标记。作为程序员，必须通过添加所需的对应代码来弥补这一问题上的缺陷。

在程序列表 8.12 尾部，还可以看到完成某些任务的两种新方法。第一个新技巧是使用&#<数字>来表示特殊字符，例如，用&#169 产生版权符号。如果想把这些特殊字符紧挨着一个普通字符放置，只须用分号将它们分开即可，例如：©1996。第二个新技巧是显示如果让别人向你发电子邮件，你的 Web 站点是什么?这只须提供一个 HREF 即可：

```
<A HREF="mailto:admin@mycompany.com">&lt;Admin@MyCompany.Com&gt;</A>
```

这里的秘密在链接的“mailto”部分中。请注意链接文本使用了特殊字符的另一种形式，&lt 与 &gt 提供了将链接文本括起来的“小于”及“大于”号。

技巧 ActiveX Control Pad 提供的 HTML 参考，在某些方面可能不全，但它提供了这些特殊字符的完整列表。查看一下主索引下的 Character Set 类即可。

添加动画 GIF

如果说前面几节中还没有介绍足够多的技术使你的 Web 站点光彩夺目的话，那么，还有许多其它你能够使用的方法。世界上的 Web 站点所有者最钟爱的方法，就是使用动画 GIF，一个动画 GIF 所做的事，就是把几幅图片打包成为一个文件，浏览器则一幅一幅地连续放映这些图片，形成连续动画的错觉。还能够使用特殊效果，来创建使用 GIF 的滑动演示，这种方法带来的问题是下载时间——滑动演示会造成用户的下载能力相当紧张。

注释 本节教你如何使用 Alchemy Mind Works 的 GIF Construction Set 来创建 GIF 。这个软件可从几个地方下载，最好的地点是直接从供货商那儿下载：

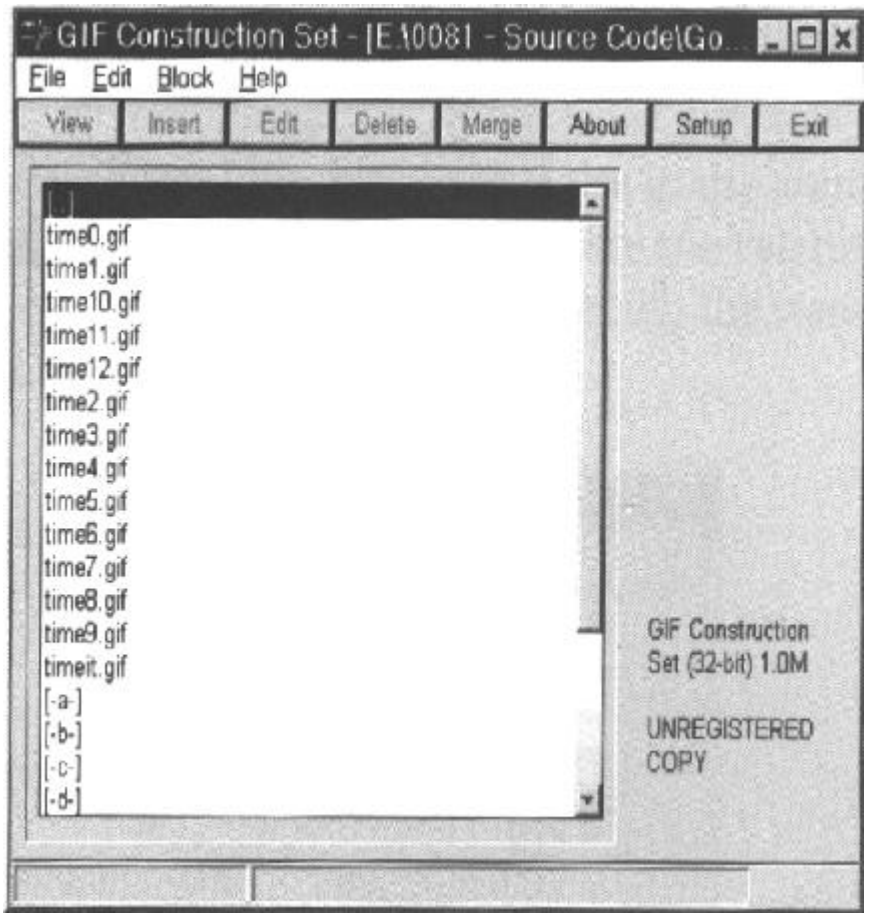
http://www.mindworkshop.com/alchemy/gifcon.html。还可以从本章前面提到的动画 GIF 观看站点中下载：

http://www.wanderers2.com/rose/animate .html。

本例中使用 GIF Construction Set 的理由有两条。第一，它是共享软件，你可以把它从 Internet 上整个地下载回来，然后用到本例中。第二，它确实是一个相当好的程序，大多数人发现用它创建动画 GIF 非常好。但是你会注意到，这个程序缺少一个实际的画图程序，但 Windows 已经提供了 Paintbrush 或 MS Paint。

如果画图程序不直接支持 GIF 文件格式(Paintbrush 及 MS Paint 都不支持），则还要有一个图形转换的实用程序。Alchemy Mind Works 的 Graphics Workshop 和 JASC.Inc 公司的 Paint Shop Pro 都是出色的图形转换程序。这两家厂商都提供产品的共享软件版本。可以在前面注释中提供的 Internet 站点中找到 Alchemy Mind Works 的产品，JASC 产品则出现在各种 BBS 和 CompuServe 论坛上（你读到本书时，也许它们也有了自己的 Internet 站点）。

打开 GIF Construction Set ，首先可以看到的是 File|Open 对话框，如下图所示：



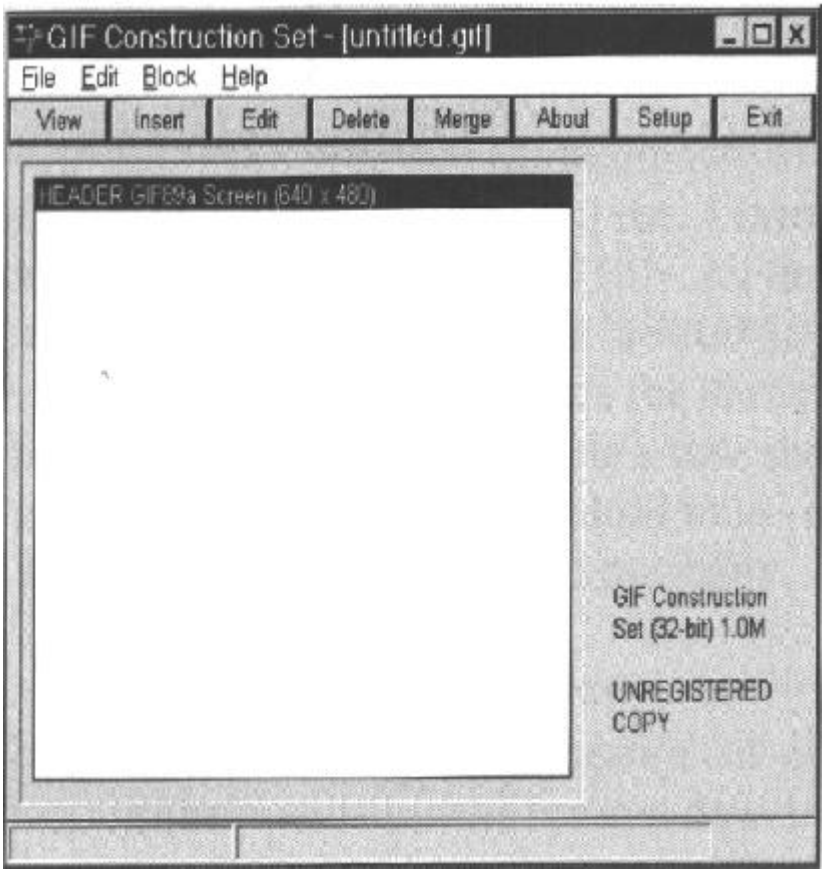
注释 和本书中所有其它的源代码材料一样，可从 http://www.osborne.com 这个 Osborne Web 站点下载 TIME.GIF。

请注意，该目录下已经有了几个 GIF 文件。TimeIt.GIF 是你在图 8.11 中所看到的静态文件，Time0.GIF 是基文件——创建动画效果使用的空白文件。创建动画时先创建这样一个空白文件，可以节省大量的时间。事实上，卡通作家也使用这一技术！他们把一个动画的公用元素一次性地画在各个单页上，然后，将它们组合起来就构成动画。只有那些唯一的项才需要每次都画。Time1.GIF

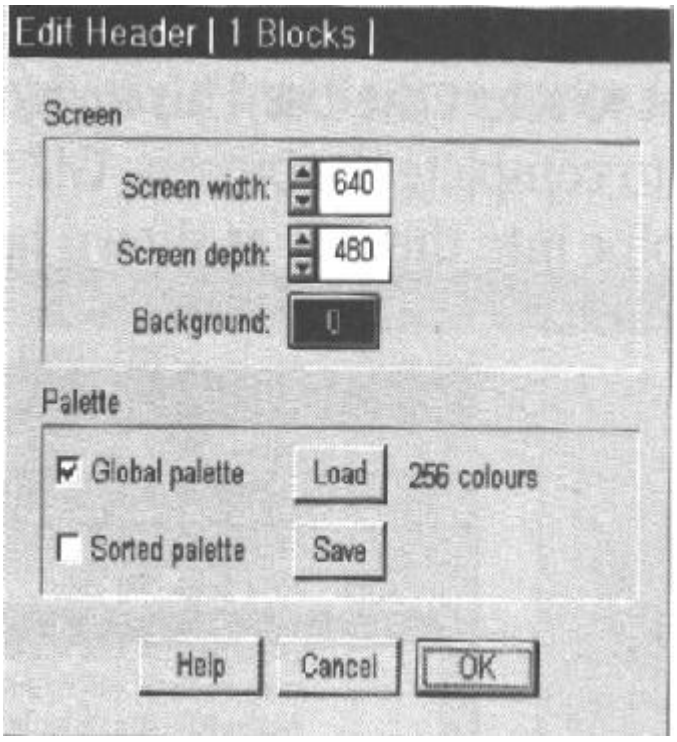
到 Time12.GIF 是实际的动画文件，把每一个看成一个动画单元好了。

让我们用这些动画单元文件来制作动画 GIF。下列操作过程并不意味着要把你束缚到一个特定模式上，但它确实说明了使用 GIF Construction Set 制作 GIF 的一种方法。

1. 用 File|New 命令创建一个新的 GIF，你将看到如下图所示的空白 GIF 对话框。GIF Construction Set 总是假定使用标准的 640×480 像素的显示区。我们需要改变其大小。



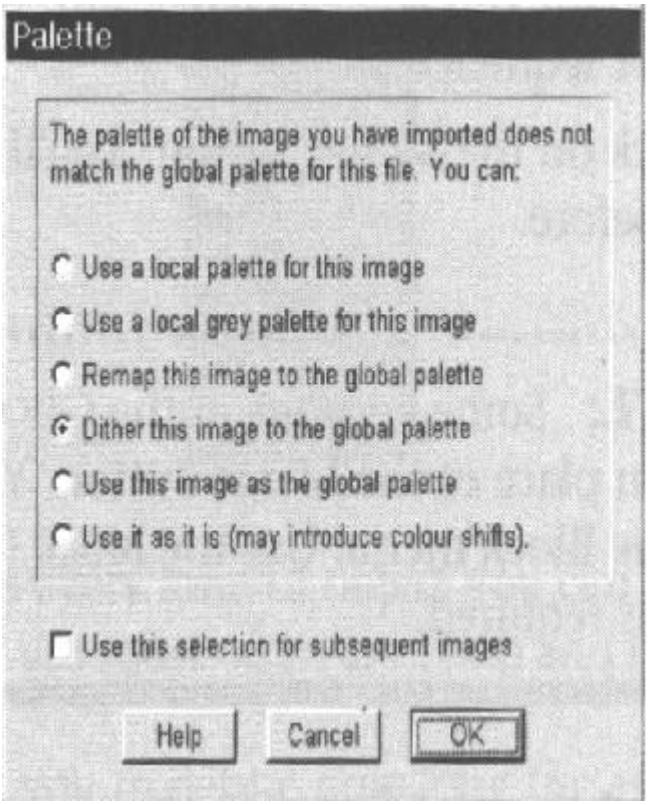
2. 双击 Header 项，就可以看到 Edit Header 对话框，如下图所示。这个对话框允许改变与 GIF 有关的特征，比如说它的大小。



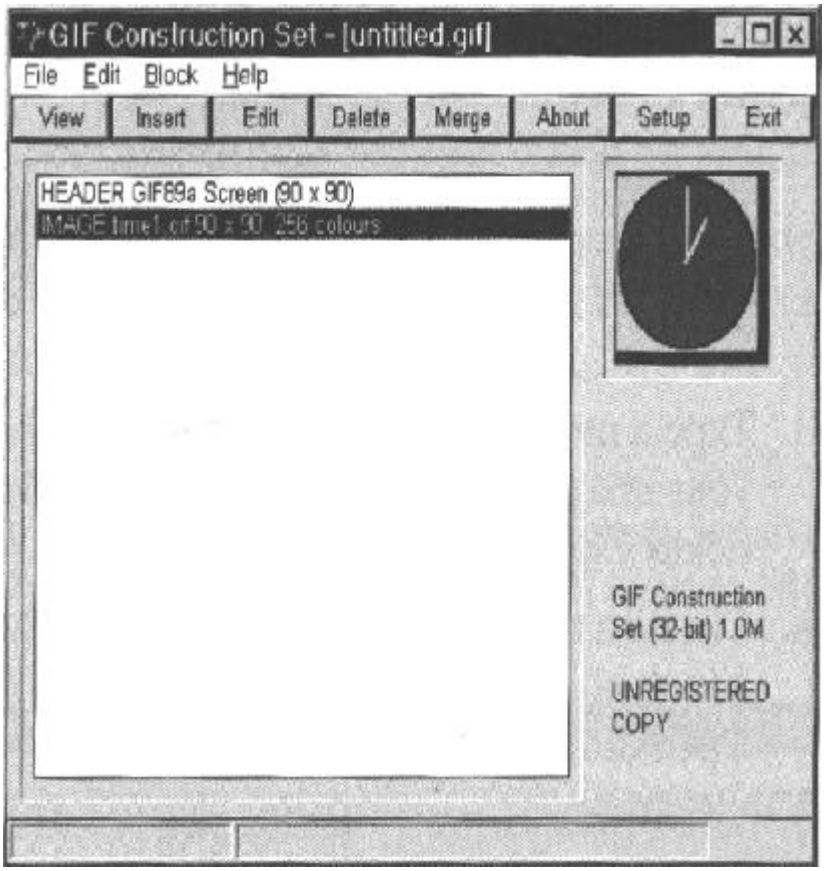
3. **在宽度和高度域中，键入新的大小，使之与图像的大小相匹配。例如，在宽度和高度域中都键入** 90，**单击** OK **使改变生效并保留下来。**

4. **单击** Merge **按钮(或用** Block|Merge **命令)，它使你可以向** GIF **中添加一幅图像，这时将看到标准的** File|Open **对话框。**

5. **双击你想在动画中使用的第一个文件，在本例中，双击** Timel.GIF，**将会看到** Pallete **(调色板) 对话框，如下图所示。用于本图形的调色板与** GIF Construction Set **使用的标准调色板并不匹配。**



6. **因为这个动画中所有图像使用同一个调色板，所以选择“** Use This Image as the global pallete **（把这一图像作为全局调色板）”选项，然后单击** OK **就完成了这一工作。** GIF Construction Set **将把一个新的图形插入到** GIF **中，如下图所示。**



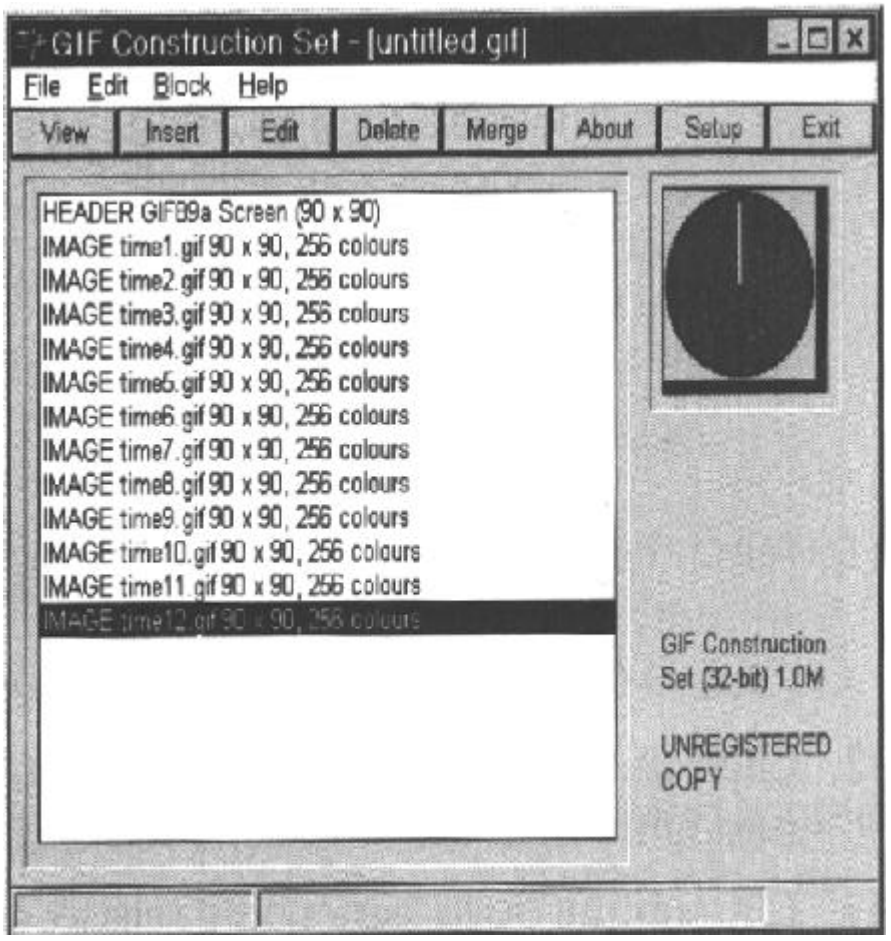
如果单击 Image **项，就会看到图像的实际副本。此时，我们无须关心该软件的这一功能；但是如果想确保你的动画文件的顺序不乱，你迟早就会用到它。**

7. **单击** Merge **按钮，将看到与前面介绍过的** File|Open **对话框相同的对话框。**

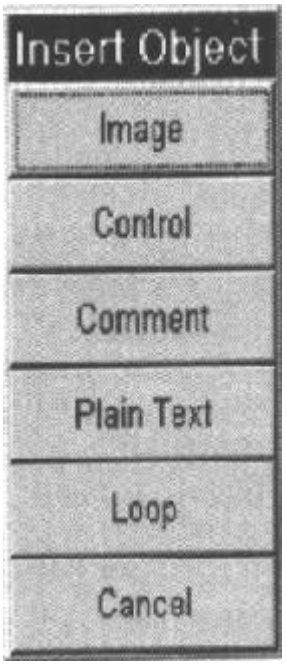
注释 GIF Construction Set 的某些版本在 Merge 按钮的位置上是 Manage 按钮。但你仍能在 Block 菜单中找到 Merge 选项。在需要用 Merge 按钮的地方使用 Block|Merge 命令，即可达到相同目的。

8. **顺次选择下一图像，并单击** OK，**若** GIF Construction Set **再次询问调色板设置，则单击** OK 。GIF **构造器则自动地把这个图像放入动画序列的下一位置。**

9. **对这一动画剩余的** GIF(Time2.GIF，Time3.GIF **等等)，重复** 7、8 **两步。完成后，对话框如下图所示。请注意这** 12 **个图像的次序。现在要插入一些控件，以使得图像正确地工作。**



10. **单击** Insert **按钮，看到** Insert Object **（插入对象）菜单，如下图所示。**



这一菜单包含了可以插入到动画 GIF 中去的各种对象。使用最多的是 Loop（循环）和 Control（控制）。

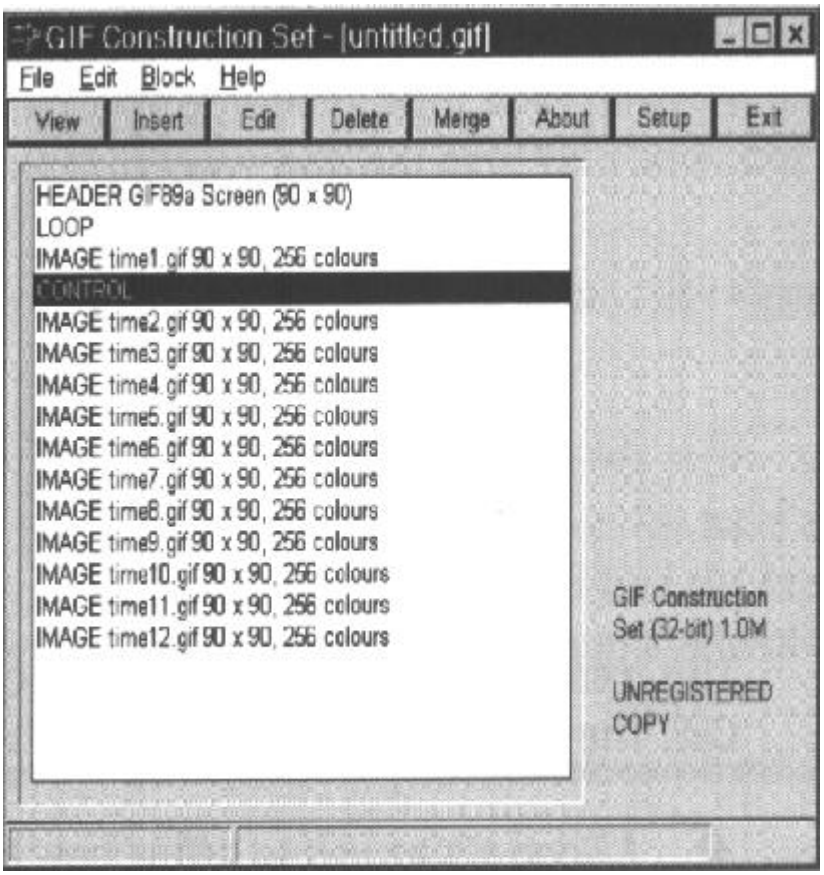
- Loop（循环）定义动画连续播放时用多长时间，Loop 对象的合理使用能制造出一些相当有趣的效果。不幸地是，许多浏览器不理睬这一特定项，所以你应该尽量少用它。（若浏览器忽略 Loop 对象，则它只是简单地保持动画 GIF 永远循环下去。）
- Control（控制）能改变动画 GIF 的行为，例如可以使用一个 Control 设置图片间隔时间。
- Image（图像）项的含义是相当明显的，打算加到动画 GIF 中去的每一幅图片都是一个 Image。使用注释将动画 GIF 的行为作成文档，也是必要的，这在你允许他人使用这个动画 GIF 时显得尤为重要。

◆ Plain Text（明文）很简单，这种文本将在动画中作为一部分显示出来。

11. 在 Insert Object（插入对象）菜单选择 LOOP（循环），GIF Construction Set 将自动地把它放到 Header 项之下。然后，在每两个图片之间放上 Control 对象，这样可以安排动画序列的速度了。

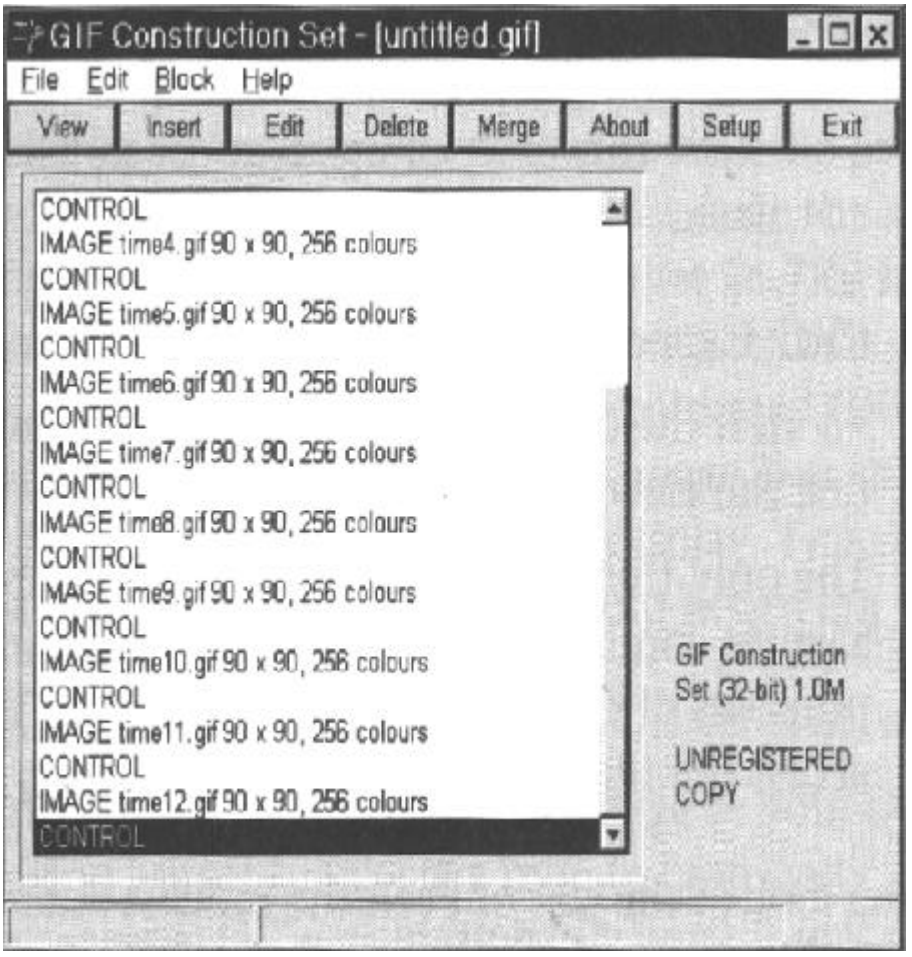
12. 单击第一个 Image（图像）项，正常情况下，GIF Construction Set 把下一个项放在你单击的项之后。

13. 单击 Insert 按钮，从 Insert Object 菜单中选择 Control ，则会看到一个 Control 项添加到了列表中，如下图所示。

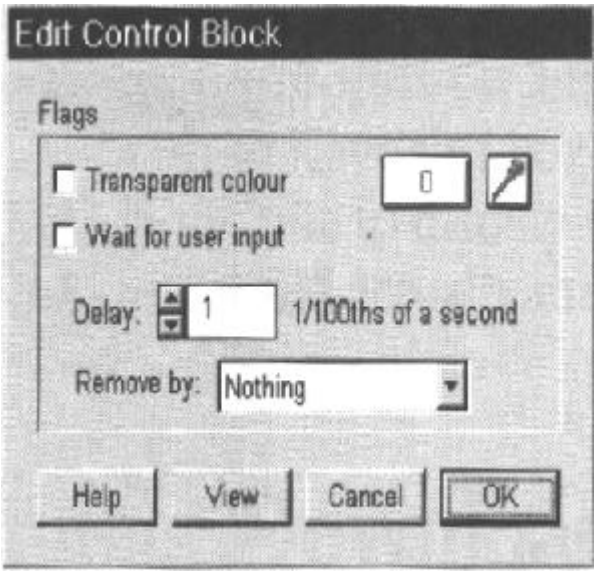


14. 单击下一个 Image 项。

15. **对每一个** Image **项重复** 13、14 **步。这些都完成后，对话框则如下图所示。（在最后一个** Image **项后，一定也要加入一个** Control **对象，因为动画** GIF **会自动地循环到第一个图像去。）**



16. **正常情况下，你可以发现使用图片间的缺省时间间隔时动画显示得非常好。但是，还是让我们看一看改变设置时需要做些什么。双击最后一个** Control **对象项，可以看到** Edit Control Block **对话框，如下图所示：**



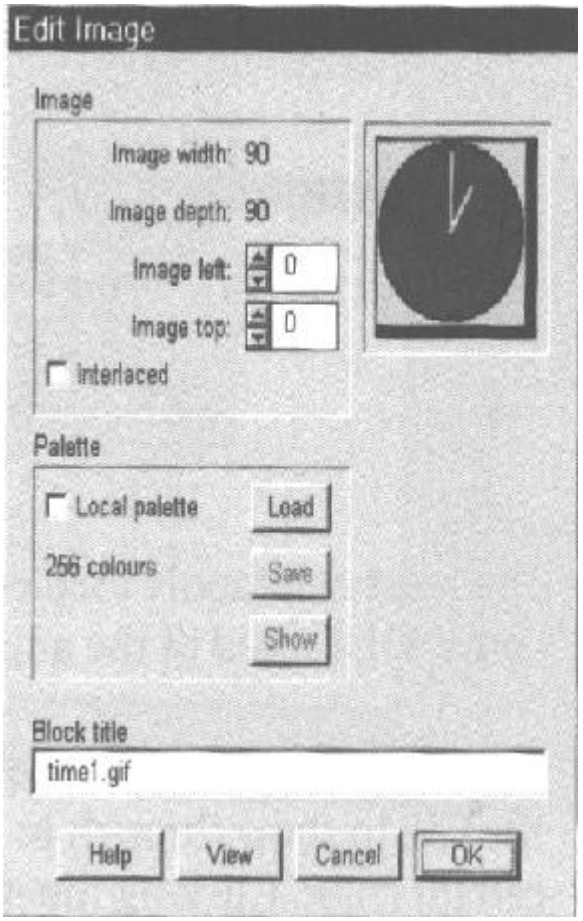
- **最常用的项是** Delay**（延迟）域，使用它可以控制动画的速度。**
- Transparent Color**（透明色）复选框的功能为，使得在** GIF **之下出现的内容，都在使用某种特殊颜色显示的** GIF **所占区域中显示出来。单击该按钮，则可以选择透明色。**
- Wait for User Input**（等待用户输入）复选框，告诉动画在这点上暂停，等待用户输入某种信息。通常是用户击一个键。**
- Remove by**（用..... 删除）域允许你确定：一旦浏览器显示这一动画单元之后，对它做些什么。请确保不要随便改动这一项，原因在于改动这一项后，在某些浏览器上会产生不可预测的结果。**

17. **单击** Cancel**（取消）关闭对话框。**
18. **为了观看整个动画，单击** View**（观看）按钮。键入** ESC **键，退出浏览区。**
19. **唯一还没完成的事是保存动画** GIF **文件。使用** File**（文件）**|Save**（保存）命令来完成即可。可以用任意文件名，但本例把它保存为** TimeIt.GIF**，随后的**

HTML 页面中我们还要访问它。

如果看一下本例中创建的这种有些简朴的动画文件的大小，就会发现它也已经用了大约 16.7KB 的内存。这差不多是这一页内存预算的三分之一，这时应该考虑一下，动画增加这么多的内存负担是否值得?

幸运的是，还有一种方法处理这一问题，双击任一个 Image（图像）对象项，则可以看到如下图所示的 Edit Image（编辑图像）对话框。



请留意，Image Left（图像左端）和 Image Top（图像顶部）域，它们允许你为动画中每个图像选择一个起点。确实不必要在每次动画单元改变时都显示

整个时钟。必须要做的仅是将前一个时针位置覆盖掉，并用系列中的下一个时针位置代替。对每个图像进行剪裁，使得前一个时针被覆盖且新的时针显示出来，这样能把动画 GIF 的大小减少 75%，换言之，16.7KB 的文件将减少到 4.7KB，需要你做的事仅仅是用画图程序剪裁图像，用刚讲过的程序将图像插入到动画 GIF 文件中，然后，使用 Image Top 和 Image Left 域，将裁过的图像正确地放到显示区中去。这个对话框还能为图片选择新的调色板，并给它起个不同的名称。

技巧 除非动画 GIF 非常大，或者你的下载预算非常小，否则不必花时间对每个图像都单独地进行剪裁。例如，这里的样本 GIF，只要它是页面上唯一的图像，就会工作得很好。

InterLaced 复选框也非常有趣。如果这个 GIF 设计成静态图像，就应该选中这个复选框。交错图像每次显示一行，这也是在大多数站点下载大图形时看到的效果。使用交错图形，可以在下载过程中给用户一点可视反馈，让它们知道没有死机。对于动画 GIF，则不要选这个框，因为显示交错图像用的时间更多。本例如果选择了交错复选框，实际上会使动画看起来好象是指针在跳跃。

8.7 使用 DIANTZ.EXE 创建组件下载(CAB)文件

前面谈到过将 ActiveX 控件放入 CAB 文件后，可以节省用户的时间。这种文件格式使得在一次下载中，可以传送多个文件并同时完成文件压缩。创建 CAB

文件不一定十分困难，但是，在投入产品环境中使用之前，一定要彻底测试它。

注释 你可能会看到，提到 DIANTZ.EXE 文件时，根据你是在看什么资源，还会用另外两个名称中的一个来称呼它。Visual Basic 用户把这个实用程序称之为 MAKECAB.EXE。ActiveX SDK 的一些版本则有一个名为 DIAMOND.EXE 的实用程序，无论你使用的实用程序名称是什么，目的却都是创建 CAB 文件，你能用它们来发布你的应用程序。

第一步是确定要把哪些文件发送给用户，例如，用户在使用其它程序时，也许拥有了全部 MFC（Microsoft 基础类库）文件。这样就不要让他们浪费时间一遍又一遍地下载这些文件。在大多数情况下，要把自己的 Web 站点上的文件都限制为独一无二的。

第二步，创建 INF 文件。包括这个文件的理由有下列几条。第一条是它能包含安装指令，从而使用户的工作更容易，而且也会减少你的支持电话。另一条是它包含了一些公用文件，ActiveX 控件需要这些文件，但不一定要作为 CAB 文件的一部分下载。你能够预先把下载这些文件的指令包括进去，当用户确实需要这些文件时，就可以从你的 Internet 站点上下载它们。浏览器中的 Internet 组件下载服务部分，并不懂得所有的 INF 文件规范，它只能使用其标准元素的一个子集，表 8.2 列出了其中可用的元素以及运用它们时的使用顺序。

表 8.2　Internet 组件下载服务使用的 INF 文件格式

项目	描述
[Add.Code]	
<Filename1>= <Section-Name1> <Filename2>= <Section-Name2> <Filename n>= <Section-Name n>	[ADD.Code]节提供了要安装文件的完整列表。它并非包含了 INF 文件中的全部文件，原因在于你并不希望把 INF 文件安装成最小模式。项目的<Section-Name>部分告诉因特网组件下载服务到什么地方寻找包含在特定文件中的安装指令。（参见下节）
[Section-Name1]	
Key1=Value1 Key2=Value2 Keyn=Valuen	文件中的每个特定节都由一个或多个关键字组成，这些关键字就象以前在 INI 文件中使用过的关键字一样，跟随其后的元素则指定了 Internet 组件下载服务能理解的关键字的值
File=[<URL>\| ThisCAB]	这个关键字指出，是从 Internet 的一个具体地址下载文件，还是从这个 CAB 文件中提取文件。对于 ActiveX 控件需要但未包括到 CAB 文件中去的那些文件，使用这一关键字，可以定义出它们的位置。通常情况下，这些附加支持文件存放在 Internet 服务器中

续表

File Version=<a>,,<c>,<d>	这个关键字指明了某文件可接受的最小版本号。如果不指定一个值，则 Internet 组件下载服务认为任意版本都可以。每个字母指定版本的一个层次，所以若控件版本为 1.0，则使用 Version=1,0,0,0
File-[Mac\|Win32]-[x86\|PPC\|Mips\|Alpha]=[<URL>\|IGNORE]	这个关键字让你对各种平台所需的支持加以区分。先定义操作系统，然后是 CPU 类型。<URL>参数则可以指定一个文件位置。IGNORE 参数则告诉 Internet 组件下载服务，该文件对于指定的平台并不是必要的
CLSID=[<Class ID>]	这个关键字让你定义文件的类标识符。你没有必要实际创建 CLSID，因为 Visual C++根据 Microsoft 研究的一个复杂公式自动地完成这一步工作。从两个地方可以得到这个值：C++源代码和 Windows 注册表。为了在 Windows 注册表中找到它。只要打开注册表编辑器，用 Edit\|Find 命令，在 HKEY_CLASSES_ROOT\|CLSID 关键字下，寻找 OCX 文件名即可。第二章示例中的 CLSID 的注册项是{D8D77E03-712A-11CE-8C70-00006E3127B7}。C++源代码的位置在控制文件中，例如，第二章中的源代码文件是 OCXEXMPLCTL.CPP。可以用这样的调用找到标识符： ////////////////////// 初始化类工厂和指南

续表

	IMPLEMENT_OLECREATE_EX(COCXEXMPLCtrl,"OCXEXMPL.OCXEXMP-LCtrl.1",0xd8d77e03,0x712a,0x11cf,0x8c,0x70,0,0,0x6e,0x31,0x27,0xb7) // ActiveX 控件的 CLSID 值不会改变（除非对于像名称这样非常基本的东西进行改变——代码改变无效），所以只需找到这个值一次即可

创建 INF 相当简便。确实需要做的是，你要想清楚需要包括什么，以及它们的位置。程序列表 8.13 展示的是安装两个文件的典型 INF 文件。第一个文件 OCXEXMPL.OCX，实际存放在 CAB 中。用户还需要 MFC40.DLL，但有可能已经安装到客户机中了。例子中包括了在 Internet 服务器上找到这个文件的一个站点，只有在客户机上没有安装这个所需文件时，这个站点才会有用。创建了文件列表和 INF 文件后，还要创建 Diamond Directive File (DDF)，DIANTZ.EXE 创建 CAB 文件时要用到它。这个文件的格式不难理解，程序列表 8.14 是创建 DDF 的一个典型例子。

程序列表 8.13

```
;OCXEXMPL.OCX 的 INF File
[Add.Code]
OCXEXMPL.OCX=OCXEXMPL.OCX
```

```
MFC40.DLL=MFC40.DLL

[OCXEXMPL.OCX]
File=thiscab
Clsid=[D8D77E03-712A-11CF-8C70-00006E3127B7]
FileVersion=1,0,0,0

[MCF40.DLL]
File=http://aux/files/MFC40.DLL
FileVersion=4.0.0.5
```

程序列表 8.14

```
;Diamond 命令文件，生成变量类型错误的完全列表。
.OPTION EXPLICIT
;定义 CAB 文件的名称
.SET CABINETNAMETEMPLATE=Activex.CAB
;创建一个 cabinet
.SET CABINET=ON
;压缩这个文件
.SET COMPRESS=ON
;文件列表
```

Install.INF

File1.OCX

file2.OCX

ù

一旦做到这一步，创建 CAB **文件就不困难了，只需使用下列命令行即可：**

DIANTAZ /F MY.DDF

第9章　JavaScript概览

如果你认为在Web站点上使用ActiveX控件可以摆脱脚本，那么，就请忘掉这个不切实际的想法吧。事实上，ActiveX控件比起其它种类的Web接口元素来说，更加依赖于脚本。理论上，你可以在没有脚本的情况下把一个ActiveX控件放进一个页面，它能在那儿挺美观地呆着，但也仅此而已。事实上，在第10章中，我们将会看到，即使你还没有为一个ActiveX控件写出脚本，它也可以执行某些功能，这些功能仅限于使用Visual C++写出这个控件时，指定它要完成的事。换言之，控件在没有脚本时是内部激活的，而脚本却可以让它与外部世界打交道。

幸运的是，除了使用ActiveX控件需要脚本语言外，使用脚本语言还有一些好处。对你键入的数据，找出其不完整或有点错这样一些小事，要靠CGI脚本在主机（Internet服务器）上完成，这时，你是不是曾经坐在那里傻等？许多人都曾有过这样的经历。由于VBScript和JavaScript都是在主机（这时是用户的机器）上执行，可以发现，用户一打入信息，马上就能把有效性验证工作完成。这节约了网络带宽，并且做到了即时向用户反馈。任何使用户高兴的事，也要受限于节约你一些时间这件事。

注　使用脚本可以节约网络带宽，而且可以使页面上的ActiveX控件与外

部世界打交道。

显然，如果你打算要花些时间使用脚本语言，那么就值得去了解一下这个舞台的主角。某些脚本语言比另一些要更流行一些，现在，VBScript 和 JavaScript 看起来都引起了大部分媒体的注意（Javascript 的 Microsoft 版本称为 JScript，我在全书中都使用 JavaScript 表示任何形式的脚本语言），JavaScript 可以在 Internet Explorer 和 Netscape Navigator 上工作，而 VBScript 只能在 Internet Explorer 上工作。（用于 Netscape Navigator 的 NCompass ScriptActive 插件仅能完成让这种浏览器支持 VBScript 这类事情。）ActiveX Control Pad 目前支持 VBScript 和 JavaScript。可以发现，市场上可以见到的大部分其它工具也支持它们。

注释 尽管你可能要选择 VBScript 或 JavaScript 与 ActiveX 控件一起使用，但本章将集中讨论 JavaScript。做出这种抉择的理由有三：第一，JavaScript 比 VBScript 在语法上更接近 Visual C++，所以，你的学习曲线的陡峭度就小（即学习起来难度小）；第二，JavaScript 天生就被两大主要浏览器 :Internet Explorer 和 Netscape Navigator 支持；第三，最新的 Microsof 的出版物及商业杂志之类的其它来源已经在说明 VBScript 正在失宠，而 JavaScript 却受到喜爱。

我们再看一下 ActiveX Scripting，它是一种 OLE 通信技术，而不是脚本语言。一旦 ActiveX Scripting 可以使用后，就能得到它是涉及到什么技术的概要，从而使你准备使用 ActiveX Scripting。总起来说，可以发现 ActiveX Scripting 依赖于目前的 OLE 技术，它加入了一些新的界面元素和一些从其它地方借鉴来

的功能。这项技术的实际面目还有点复杂。（无疑我们将会看到这些新的界面元素。）

技巧 如果计划采用 ActiveX 控件或本章中讨论的任一种脚本技术、耗费大量时间去开发 Web 页的话，你肯定需要得到一种比 ActiveX Control Pad 功能更强的工具。诸如 Microsoft Visual InterDev 之类的产品，则使你不必对基本的 HTML 标记了解太多，而且很容易地创建 Web 页。Visual InterDev 还使你可以在 Web 页内使用 ActiveX 控件，ActiveX Control Pad 对于实验用途是最有用的，使你可以创建包含脚本的小小的 Web 测试页面。

本章用大部分时间来考察 ActiveX 控件和脚本怎样配合起来进行工作。让我们从一个简单例子开始，这个例子使用了 ActiveX Control Pad 提供的一些控件。例如，也许你已经注意到，我们不能通过把下拉列表框粘贴到页面上并通过访问属性对话框来定义下拉列表框的列表项。道理很简单：需要使用脚本在运行时添加列表项。本章各节均讨论 JavaScript，但是，也可以使用 VBScript 完成类似的任务。

走过打基础阶段后，我们将用些时间讨论一些中等难度的脚本例子。例如，讨论一下怎样使用脚本与已建立的控件打交道。虽然本章缺少一个实际上更复杂的控件来演示深入的概念，但我们依然研究了足够多的基本概念，说明了如何在复杂情况下使用脚本的方法。

9.1 ActiveX Scripting概要

目前，ActiveX Scripting还不能像我们谈论过的VBScript和JavaScript语言那样，成为一种语言（宏或其它形式的语言，本章下一节将讨论JavaScript语言）。最好将它定义为客户机与服务器之间的一种通信方法。实质上，它是Microsoft为OLE自动化起的新名称。说OLE自动化与ActiveX Scripting完全相同也许不够准确，因为Microsoft没有把ActiveX Scripting的能力停留在应用级别上。（大多数Microsoft产品和诸如CorelDRAW之类的一些其它产品，在应用级别上支持OLE自动化!）ActiveX Scripting还包括了Internet。

可移植性问题 本书前面的例子中，MFC AppWizard（应用程序向导）提供了OLE自动化复选框，这一点也许你已注意到。这一复选框，像诸如CorelDRAW之类的产品一样，向你的应用程序添加相同种类的脚本能力。换言之，可以在本地的机器或网络中使用脚本能力。但是，这一能力仍然不是完整的ActiveX Scripting，因为它不包括Internet。你仍然需要手工地向Visual C++应用程序添加这种功能。因为ActiveX Scripting标准还没有最后定下来，所以你看到的仍然是进展中的东西。

既然ActiveX Scripting不是语言，而是通信规范，那么，脚本语言，包括语法，这一类问题就留给了脚本供货商来提供。这一标准的目的在于定义一种方法，使脚本主机可以调用各种脚本引擎，并且用于OLE容器中对象间的通信。

从实质上看，脚本更像是可执行代码的大杂烩，它可能含有文本、伪代码，甚至还有原始二进制格式的可执行代码。ActiveX Scripting 的实质是，它是基于 OLE 的、可用于应用程序和 Internet 的通信媒介。

WEB 链接 写作本书时，ActiveX Scripting 说明书仍在变动之中。如果打算用 ActiveX Scripting 作为其它技术的替代或补充，也许要用些时间看一下最新动向。进而，本节并不完全涵盖 OLE 主机和脚本引擎需要的所有接口方法的调用语法。欲了解目前 ActiveX Scripting 说明更多的信息，请查看 http://www.microsoft.com/intdev/sdk/docs/olescrpt/ 要查找有关 ActiveX Scripting 主机的信息，下列两个站点不错：http://www.microsoft.com/msdn/sdk/inetsdk/help/wsh/wobj_2.htm 和 http://www.microsoft.com/msdn/sdk/inetsdk/help/compdev/scripting/hosts.htm。如果想多了解一些关于建立一个 ActiveX Scripting 引擎的需求，请查看：http://www.microsoft.com/msdn/sdk/inetsdk/help/compdev/engines/engines.htm。

让我们花点时间定义几个术语。首先，精确地说，什么是脚本主机呢？它是支持脚本引擎的一种应用程序。定义脚本时，是脚本主机接受这个脚本并向引擎发出命令。ActiveX Scripting 主机的最通用的例子是 Internet Explorer 3.0。使用插件可以向 Netscape Navigator 添加这种能力。第 8 章讨论过一种插件

——NCompass ScriptActive。脚本引擎，当然不限于这些产品中的任何一种。你可以在一个定制的浏览器应用程序中建立它，或者在一些应用程序中找到它。例如，很有可能，在不远的将来，Microsoft 会使其全部应用程序产品（如 Microsoft Word）都符合这一标准。

一个 ActiveX 脚本引擎是一个对象，这个对象才真正地对脚本进行解释。对精确的语言语法甚至脚本的形式都没有限制。理论上，写 ActiveX 脚本，可使用任意语言，包括 VBScript、Java 和 JavaScript 。和本节后面要讨论的任何一种 OLE 对象一样，一个 ActiveX 脚本引擎提供的是特殊接口。另外，可以使用 OLE 自动化包装工具写出 ActiveX 脚本引擎（Microsoft 计划为 ActiveX Scripting 提供一种特殊形式的 OLE 自动化包装器）。使用包装器格式的好处是，它保持了代码的短小精炼——这对于浏览器和在线脚本引擎都是理想选择。不利之处是，如果直接写出 ActiveX Scripting 接口，你将失去对运行时你可以有的名字空间、持久模型以及其它著作元素的控制。

ActiveX Scripting 要求有四种新的接口元素。可选项 IActiveScriptSite 和可选项 IActiveScriptSiteWindows 接口为主机专用，而 IActiveScript 和可选项 IActiveScriptParse 接口则为脚本引擎专用（若不实现 IActiveScriptParse，则需要实现某种形式的 IPersist 接口）。每个接口元素执行下面列出的一种具体功能：

IactiveScriptSite 这个接口的主要目的是为 ActiveX Scripting 引擎创建一个站点。这与其它的 OLE 实现中的容器的概念(idea)相对应——容器容纳的是诸如 ActiveX 控件之类的东西。因为这是一个容器接口，所以它监测脚本的开始、终止及脚本出错等等一类的事件。

IactiveScriptSiteWindow 任一个提供用户接口的 ActiveX Scripting 主机，

也需要提供本接口（服务器不一定应用它）。实际上，如果需要用于脚本目的的窗口，则也需要实现这一接口。IActiveScriptSiteWindow 提供了两种主要方法（尽管可以提供其它所需的方法）：GetWindow()用于创建窗口，EnableModeless() 则设置窗口的模式(modal)条件。GetWindow()的功能类似于 OLE 自动化专用的 IOleWindow::GetWindow()函数，而 EnableModeless() 的功能则类似于 IOleInPlaceFrame::EnableModeless()函数。

IactiveScript　这一接口使你与脚本本身一起工作，用它可得到脚本的站点、启动和终止脚本、使用脚本项目、关闭脚本、或者建立线程状态与参数。

IactiveScriptParse　这一接口从脚本主机接受脚本，它使你能创建一个新脚本、 向一个现存脚本中添加原始脚本片段或对一个现存脚本进行分析。

Visual C++提供了一个方便的实用程序，称为 OLE/COM Object Viewer（OLE/COM 对象浏览器），使用它可以更详细地看到这些接口（Microsoft 在其 Visual C++的最新版本中将名称缩写为 OLE View（OLE 浏览器））。本书中我们将多次用到这个实用程序，如果你还没有安装的话，应该安装上它。让我们先看一看 ActiveX Scripting issue 的主机端。和以前提到的一样，Visual C++仍然提供一个 OLE 自动化接口，这个接口不提供与 Internet 连接的功能。接着请打开 OLE/COM Object Viewer，就会看到一叠文件夹。打开 Document Objects（文档对象）文件夹，再打开 XYZ Single Document 文件夹，就会看到接口列表，与图 9.1 类似，这些接口是 MFC 实现的，建立应用程序时可使用它们。

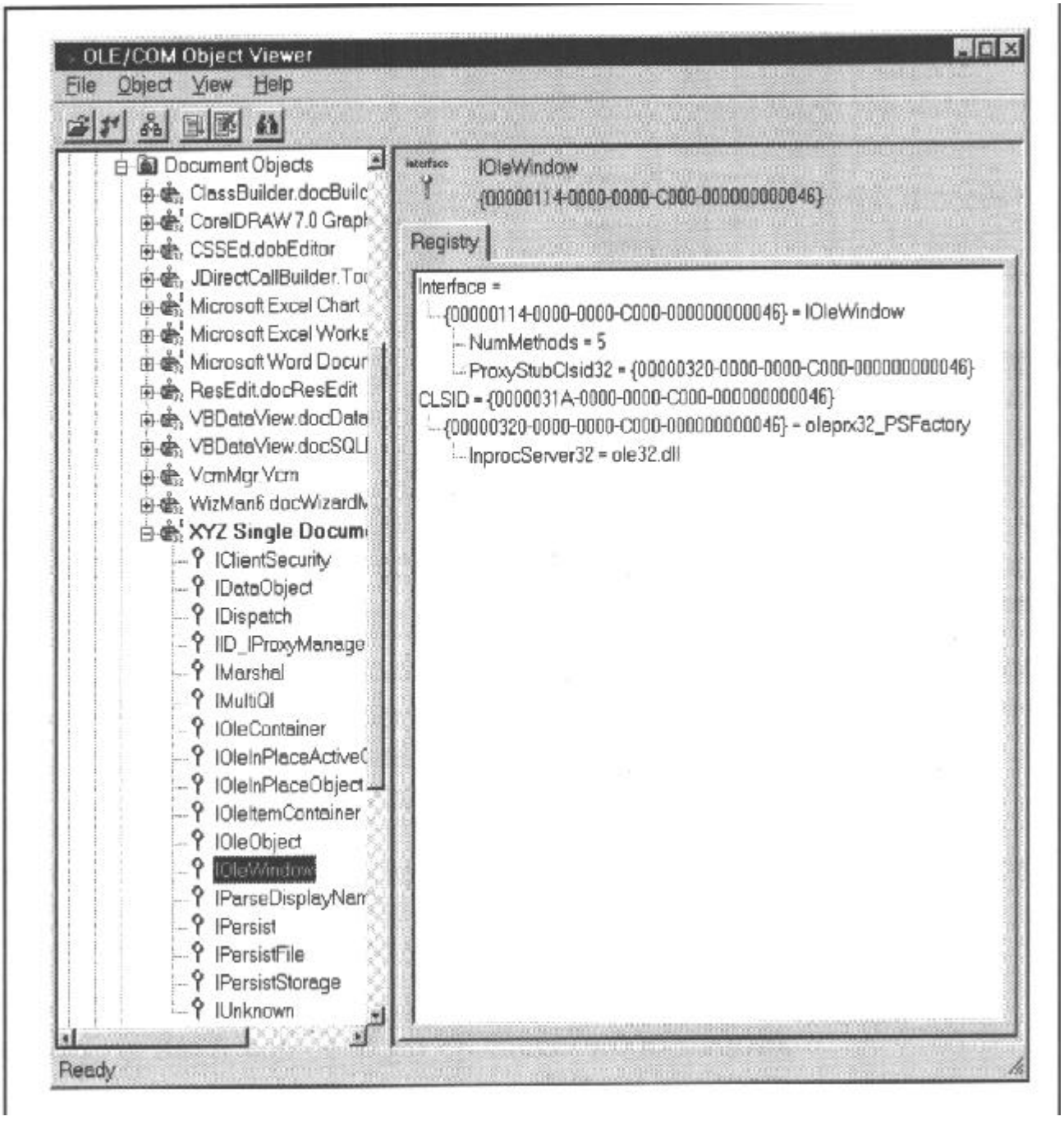


图 9.1　**在** OLE View **中查看** ActiveX **接口**

请注意在左边的窗格中的 IOle Window 接口已被选中。看一下右边的窗格，可以看到这个接口提供五种方法。进一步研究表明，这些方法之一是 GetWindow()。前面提到过，没有 IActiveScriptSiteWindow 接口，而 IOleWindow 接口却存在，这意味着你的应用程序支持 OLE 自动化，而不支持 ActiveX Scripting。现在，如果你看一下这个列表中的每个其它应用程序，（如果有的话）包括 Office97，就会发现，没有一个应用程序是支持 ActiveX Scripting 的。

警告 如果在 OLE/COM Object Viewer 中创建了应用程序或控件的实例，而后没有不释放相应实例，那么就会发生奇怪的事情。例如，你的机器会意外地死机！每当你查看一个应用程序或控件支持的接口时，就不得不创建一个实例来完成这个操作。通过查看应用程序的名称，也能够知道是否存在一个对象的实例。OLE/COM Object Viewer 用粗体显示打开的对象。为了释放你创建的实例，用右键单击对象名（比如是 XYZ Single Document），然后在上下文相关菜单中选择 Release Instance（释放实例）即可完成。幸运的是，OLE/COM Object Viewer 在你离开该工具时总是会关闭对象的实例，但是如果在查看会话期间机器的内存开始不够用时，就应该关闭对象的实例。应该记住，每创建一个实例，就要占用一些内存。

客户端对 ActiveX Scripting 也是支持的，至少在一种初步形式上是这样。释放 XYZ Single Document 的实例，然后关闭 Document Object 文件夹。接着打开 ActiveX Scripting Engine 文件夹，可以看到两个项，一个是 JavaScript，另一个是 VBScript。这两个项都是 Internet Explorer 支持的，但我们期望看到 Netscape

Navigator 也支持它们。打开 JScript Language(JavaScript)项，可以看到如图 9.2 所示的典型的接口列表（你的列表也许会稍有不同）。

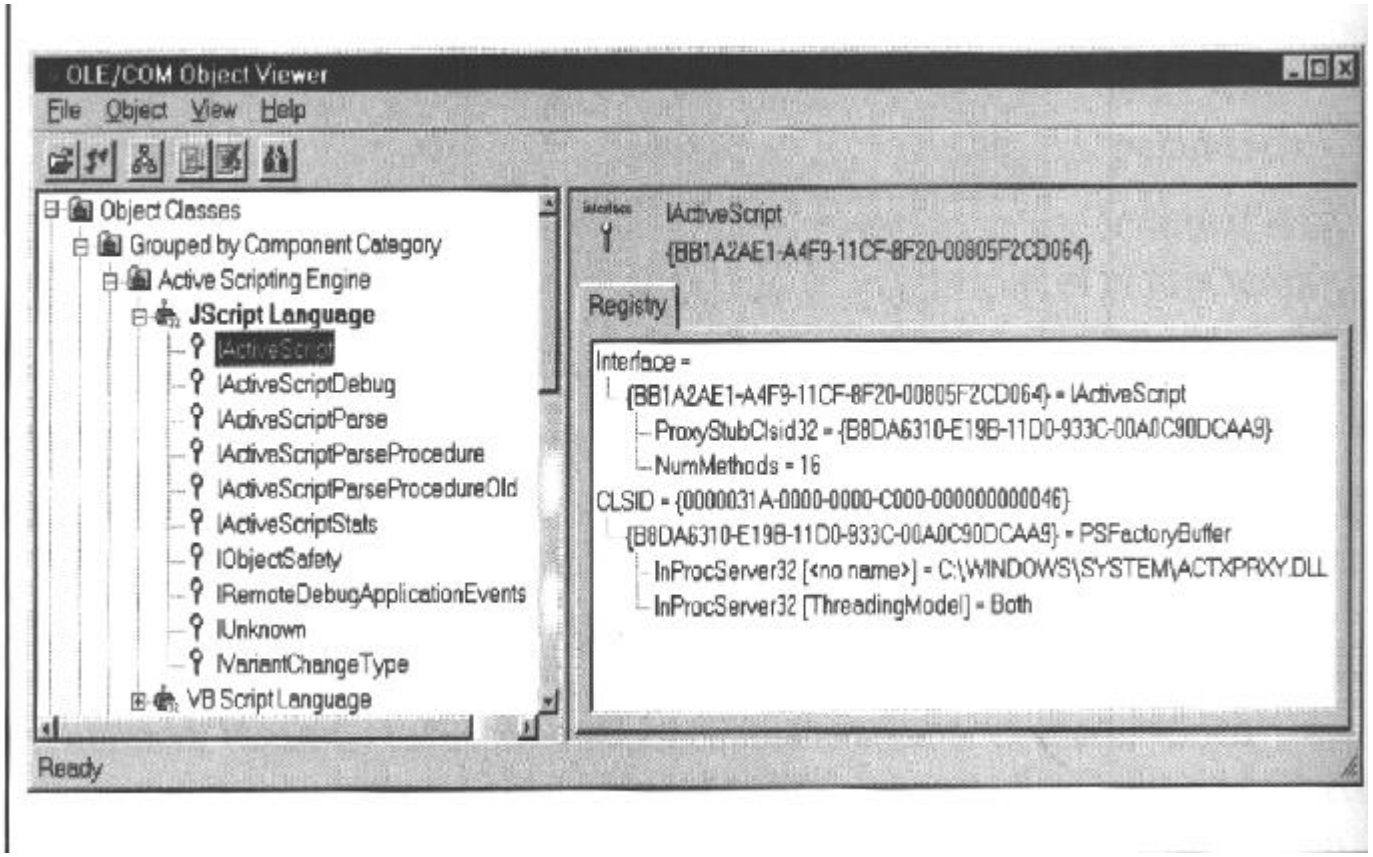


图 9.2　在 OLE 视图中查看 JavaScript

请留意被选中的项 IActiveScript，它是我前面描述过的第一个客户机端接口，在它下面，可以看到 IactiveScript-Parse 项，它是我前面描述的第二个客户机端接口。这两个元素的存在，意味着 Internet Explorer 当前把 ActiveX Scripting 做为一个客户机来支持它。但是，在 OLE/COM Object Viewer 的这一部分中却连一个相联系的服务器项都找不到。

让我们对于 ActiveX Scripting 的主机与客户机的关系再多说几句。一个 ActiveX Scripting 会话期，要涉及到三个不同的元素：主机、引擎以及包含代码和控件的窗口（页）。在这三个元素之间建立通信的过程有 8 个步骤：

1. 主机根据需要装入文档或工程。这不是 ActiveX Scripting 专用任务，因为任意环境都需要执行这一步。
2. 主机创建一个脚本引擎。通常，它要使用<OBJECT>标记中的 PROGID 属性，并通过调用 CoCreatInstance()来完成这个任务。
3. 然后装入脚本。主机完成这件事的方式有几种。如果脚本存贮为驻留(Persisted)数据，那么，主机就使用引擎的 IPersist::Load()把它再次装入内存。如果主机需要装入一个新脚本，那么它就调用 IActiveScriptParse::InitNew() 或者 IPersist::InitNew()创建一个空脚本。把脚本当成文本来维护的主机，然后调用 IActiveScriptParse::ParseScriptText()把脚本装入引擎（使用其它格式维护脚本的主机则不得不设计自己的装入方法）。
4. 主机把脚本装入后，必须用实体填入引擎的名字空间。实体可以是窗体、 页或其它文档。主机使用 IActiveScript::AddNameItem()方法完成这一任务。显然，它不必装入存贮为脚本驻留内存一部分的那些实体。低级项（比如 ActiveX 控件）需要不同的装入技术，主机使用 ITypeInfo 和 IDispatch 接口来完成这项工作。
5. 现在，ActiveX 脚本引擎已取得了它需要运行脚本所需的一切。主机发出一个 IActiveScript::SetScriptState(SCRIPTSTATE_CONNECTED)调用，这就是让脚本引擎启动脚本。它相当于标准 C 程序中调用 Main()函数。

6. 在脚本引擎真正开始运行脚本之前，它还要先干几件家务活。例如，需要为每个顶级项联系上一个符号。脚本引擎干这件事时使用的是IActiveScriptSite::GetItemInfo()方法。

7. 脚本引擎真正运行脚本之前的最后一个步骤是，在脚本元素与与对象相关联的事件之间建立适当连接。例如，如果存在一个脚本项，它处理一个按钮的onClick （单击）事件，那么脚本引擎就需要在运行脚本之前建立这个连接。脚本引擎使用IConnectionPoint接口完成这一工作。

8. 最后，脚本引擎开始运行脚本，运行脚本时，脚本引擎需要访问与HTML 页上或其它文档中的对象相关联的方法和属性。它使用 IDispatch::Invoke()或其它标准的OLE 捆绑方法来干这件事。

现在，你对脚本如何工作这一点已有了相当好的理解。让我们快速地看一下，脚本引擎运行时作出反应的几种方式。脚本引擎不只是具有简单的两种操作状态：ON/OFF，实际上它具有好几种状态，使用这些状态，可以确定需要干些什么才能使它运行。（使用 IActiveScript::GetScriptState()方法可以得到当前状态。） 以下列表列出了脚本引擎的各种状态：

- Uninitialized（未初始化） 这表明脚本引擎还没有准备好，通常，主机在脚本引擎离开这一状态前，要装入运行脚本所需的脚本和实体。
- Initialized（已初始化） 脚本已装入（通常用 IPersist ），运行站点已设定（使用 IActiveScriptSite），但脚本引擎还没有真正地与一个主机相联系。脚本引擎处于这种状态时，不能运行任何代码。一旦脚本引擎进入已启动状态，则由IActiveScriptParse::ParseScriptText()执行的代码将开始运行。

- Started（已启动） 这是一种过渡状态。从这种状态使用 IDispatch 就能运行代码了。所有代码与对象一起装入。但是，脚本引擎尚未在脚本元素与对象事件之间建立连接。如果脚本运行到需要执行事件处理的地方，它就被阻塞（停止），直至脚本引擎完成所需的初始化。
- Connected（已连接） 脚本引擎做好了去执行需要由它来执行的每项任务的准备时，就是进入了这种状态。
- Disconnected（连接暂停） 脚本引擎进入这一状态后，实际上脚本已暂停。脚本仍处于装入状态，所有连接依然存在，但脚本引擎却不准备回答主机的请求。脚本引擎可以在不丢失脚本当前运行位置的情况下，离开这一状态并返回已连接状态。
- Closed（关闭） 在这种状态中的脚本引擎，不再回答调用。你发出 IActiveScript::Close()调用后，就进入这一状态。

9.2 Java Scripting 基础知识

在深入讨论之前，我们先澄清一件事。Java 和 JavaScript 是两种完全不同的实体。如果你看到的是一种功能强大的编程环境，那么，你看到的语言是 Java(或 Mictosoft 版的 Java，称为 J++)，而不是 JavaScript。反之，如果你看到的是与 VBScript 一样外型的宏语言（尽管功能不同），而且是面向 C 的，那么这是 JavaScript。另外，Java 是 Sun 公司的发明，而 JavaScript 却是由 Netscape 的杰作。肯定是 Netscape 的市场人员造成了这种容易混淆的概念——对两种产品赋

予了相同的名称。JavaScript 原来的名称是 LiveScript。

可移植性 JavaScript 一直是浏览器大战中的受害者之一。Internet Explorer 和 Netscape Navigator 在各自浏览器中，使用类似的但又不完全兼容的 JavaScript 版本。有一个小故事说，Netscape 为它的 JavaScript 版本推出公开说明书时慢了一步，而 Microsoft 就抓住时机设计了自己 JavaScript 版本。有传言说，两个公司正在制定 JavaScript 标准的联合版本——对于这一点，我不会屏住呼吸等待它。同时，使用 JavaScript 时，你不得不面对兼容性问题。好消息是，通过努力，可以做出能在两个浏览器上都行得通的你自己的一个脚本版本。当然，你肯定需要在两种产品中测试你的脚本，从而确保他们像所预期地那样工作起来。

现在我们已把 Java 和 JavaScript 区分清楚了。我们来看一下语言本身。可以发现，JavaScript 在相当程度上与 VBScript 一样，都是用于相同的目的——作为在一个 HTML 页上用 ActiveX 控件和 Java applet 工作的一种手段。事实上，使用 JavaScript 可以处理任意对象，唯一的制约因素是弄清楚该怎样去做。对于我们在本章后面几节中要看到的许多其它目的的大部分，JavaScript 迟早会有用。JavaScript 提供了一套与 VBScript 截然不同的特色，这些特色能增强 Web 站点的表达能力，弄明白这一点才是重要的。

注 除了名称相似外，JavaScript 和 Java 实际上是两种截然不同的编程环境。

与 VBScript 不同，对于 JavaScript 来说，不存在任何与它直接兼容的版本。

把它与 C(或 Java)相比，不会使你得到更多的信息，因为 JavaScript 设计为面向 Internet 使用的具有一套独立特色的产品。使 JavaScript 在 Internet 出名的主要是它的兼容性。肯定地是，不能把 JavaScript 应用程序搬到 C 编译器去进行编译，并期望它做事。但是，你能指望它比 VBScript 能在更多的原始浏览器环境下工作。本节中我们快速地浏览一下 JavaScript 提供了些什么，然后在本章中我们可以看到它是如何工作的。

可移植性 请记住，JavaScript 使用与 C 类似的语法。这意味着在有些情况下，JavaScript 看起来有点像 C，而在其它方面，JavaScript 将使用通常不能在 C 中使用的新结构。当你在 JavaScript 环境中，能从 C 应用程序中使用一些应用程序逻辑时，不要以为不做任何的修改就能这样做。JavaScript 毕竟与 Visual C++不直接兼容。

JavaScript 处理 HTML 文档时，采用了对象映射法(approach)。它建立了对象的一种层次结构，从窗口开始，文档在窗口中，窗体在文档中，最后，对象在窗体中。使用 JavaScript，可以通过定义路径的方法与每个对象通信。例如，如果想对某个窗体做点事，你要从窗口名中分离出窗体名：Window1.Form1。注意窗口名与窗体名用一个圆点分开，许多程序员把这叫做点语法引用。

JavaScript 存在一些与 C 完全不同的内容。首先，JavaScript 是一种松散型语言。这意味着不一定要对变量进行说明。其次，JavaScript 中存在对象，但不存在类或继承。这意味着不能创建新的对象类型。但是，能够把一个现有对象类型扩充到某种程度。你使用的对象，就是 HTML 文档或窗体中包含的那些对象。例如，甚至是基本的 JavaScript 应用程序也能够访问 Window 对象——每个

HTML 文档必须有这样的对象。在 Window 对象中，可以找到缺省对象，如文档、框架、历史、位置等等。

还可以扩充对象的数目供你使用，但不要用创建新类的通常方法。本章中讨论的方法是使用 ActiveX 控件。使用 ActiveX 控件的扩充方法，依赖于控件作者让你使用什么样的属性与方法。从实质上看，一个 ActiveX 控件，属性出现于属性页中，方法就是暴露出来的事件。（若要对建立控件方面了解更多的信息，请看第 10 章中例子。）重要的是要记住，JavaScript 是基于对象的而不是面向对象的。最后，JavaScript 是动态约束(bound)的，这意味着它在运行时才核查对象。C 和 Java 则都是静态约束的，他们都是在编译时核查对象。

WEB 链接 有许多站点可以学习 JavaScript。首先是找一份语言指南，语言指南最好到 http://home.netscape.com/eng/mozilla/Gold/handbook/javascript/index.html 去找。可以发现，在那里不仅有语言指南，而且还有对各种语言元素的极好讨论。幸好你可以从 http://www.jchelp.com/javahelp/javahelp.htm 得到一份具有 Windows 帮助格式的 JavaScript 文档的教学版。

关于 JavaScript，要记住的最最重要的事情是，和 Java 一样，你在操作系统硬件方面受到严格的限制。其原因是（尽可能多的）防止出现安全缺口。不幸的是，实际能从 JavaScript 得到的保护层次是有限的（参见本节末尾高级技巧“三个普通的 JavaScript 安全缺陷”中的详细讨论）。一些人把这些限制真当成了一回事来对待，但是你能够使用 ActiveX 控件绕开这些限制中的大部分。

因为用户在下载前不得不请求 ActiveX 控件，所以安全性得到了加强。ActiveX 控件要求一定要签名，也有助于改善安全性。但不论你使用何种策略，确实想毁坏你机器的人总有办法去达到他们的目的。

遇到的硬件访问限制之一是，不能写文件。这一规则的唯一例外是能够向 cookie 中写数据。cookie 是保存在浏览器缓冲区目录下的一部分数据。cookie 的目的是，从一个会话转向下一个会话时存贮配置信息。一些程序员也把 cookie 用作其它的小型存贮目的，例如，可以用它来存贮用户到达的最后一个站点的 URL，以便于在下一会话中让用户再返回到那个站点。

JavaScript 支持四种不同的变量类型：数字、字符串、布尔值和一个特殊的 null（空）值。可以使用函数调用将一种变量类型转换成另一种。但是，你能够很容易地混合使用变量类型。例如，可以创建一个名为 nANumber，给它赋一个数值的值。就在代码的下一行，你就能够赋给这个变量一个串值，JavaScript 对此不会抱怨。还能够在同一行代码中混合使用变量类型。例如，nANumber="SomeAnswer Equals "+42 在 JavaScript 中是完全合法的。这一点是下列做法的理由之一：通过使用（引言中解释过的）匈牙利表示法强制使用你输入的某种变量。

WEB 链接 与在其它领域中进行程序设计一样，从 JavaScript 程序员伙伴那里取得反馈信息是相当重要的。如果想得到关于 JavaScript 程序设计技术相当公正的观点，Comp.Lang.JavaScript 是最好的地方之一。如果要知道 Microsoft 的看法，那么就看一下 microsoft.public.frontpage.client 或者 microsoft.public.

activex. programming.scripting.jscript。（其它的 Microsoft 新闻组偶尔也有 JavaScript 方面的消息，但从这两个新闻组可以得出大部分消息。） comp.infosystems.www.authoring.html 偶尔也会提到 JavaScript。

在 JavaScript 中，函数是使用模块化编码的唯一形式。事实上，这也是 JavaScript 实际使用类似 C 的外观的第一点。和大多数程序设计语言一样，函数调用接受参数，也可以有返回值。下面的例子是一个典型的 JavaScript 函数，请注意，它调用了 Document.Write()，这个函数是脚本开始时用得最多的方法之一，使用它不仅可以显示文本，而且如同我们这里所做的一样，也可以写出 HTML 代码。

```
function MyFunction(nSomeValue)
{
    Document.Write("The value received was: ",nSomeValue,".","<BR>")
    Return nSomeValue +1
}
```

大多语言都支持某种条件语句，JavaScript 也不例外。它使用的最简单的条件形式是：(<条件>)?<值 1>:<值 2>，使用它把两个值中的一个赋给变量。例如，lISGreater=(值 1>值 2)?true:false，如果值 1 大于值 2，那么 lIsGreater 就等于 TRUE。还可以像下面例子那样使用 If.....Else 条件语句。JavaScript 目前不支持 Case(switch)语句，这样，当核查许多不同值时，写出其代码就要困难一些。

```
if (Value1>Value2)
{
        Document.Write("Value 1 is greater than Value 2.")
        Return true
}
else
{
        Document.Write("Value 2 is greater than Value 1.")
        Return false
}
```

循环语句是所有程序设计语言都支持的另一种操作。可以发现 JavaScript **支持两种不同的循环结构：**for **和** while**。下面对每一种循环形式都给出一个例子，请注意他们看起来几乎和** C **中的等价结构一模一样。**

```
for (nCounter = 0, nCounter < 10,nCounter++)
{
        Doucument.Write(nCounter)
}

while (nCounter < 10)
{
        Document.Write(nCounter)
```

```
        nCounter++
    }
```

以上给出了 JavaScript **最基本的概貌。本章中会看到大量的** JavaScript **代码，所以不用担心你还有很多东西没弄明白。需要记住的主要事情是，**JavaScript **很像** C**，但无论你怎样富于想象，它都不是** C**（现在你该已注意到相当多的区别了）。**

高级技巧

三个普通的 JavaScript 安全缺陷

恐怕你认为脚本是很容易监测的，但是与脚本相关的许多安全问题一直存在，对于这些问题，近几个月来我在贸易新闻与新闻组上已指出过。简单地说，无论 VBScript 还是 JavaScript 都不是安全的（尽管做了大量工作以找出 JavaScript

存在的安全缺陷），在 http://www.osf.org/~loverso/javascript/ 上能够找到目前已知的 JavaScript 的安全问题。下面列出当前三种最普通的问题：

- 欺骗用户上载(upload)文件。尽管 JavaScript 不得不请求得到用户的允许来上载一个文件，但黑客可以用各种方式隐藏这一请求。黑客真正需要的全部东西，只不过是具有令人感兴趣的标题的按钮。像 Windows95 使用的那种上载口令字文件，是很容易被破译的——这使得黑客很轻易地就能闯入你的系统。
- 获取文件目录。JavaScript 应用程序不必请求得到任何人的允许就能加载你机器中的目录。事实上，它还能加载你访问过的任意网络机器的目录。一个知道你的硬盘组织方式的黑客，闯入你机器时就要容易得多了。
- 追踪访问过的站点。黑客们通过追踪你访问过的站点，可以对你了解很多。JavaScript 应用程序使得追踪很容易实现，它能追踪你访问的每个 URL 并把地址发向黑客的机器。和文件上载问题一样，用户不得不指出是否允许追踪，但黑客能够把这一允许伪装成其它东西。

9.3 使用独立的脚本

如果你熟悉 C 语言，那么 JavaScript 是相当简单的。由于在本章上节中已初步介绍过 JavaScript 语法，所以现在让我们考察一个脚本样本。这是 JavaScript 的一个典型应用程序，用于计算定货单上用户项的结果，使得服务器不必在

Internet 上来来往往的传送数据。

WEB 链接 写作本书时，Microsoft 还没有在其任一个程序设计产品中包括 ActiveX Control Pad 这个实用程序。可以从几个地方得到这个实用程序，可以由你的 MSDN (Wicrosoft 开发者网络)预订处向你供应这个实用程序，还可以从 http://www.microsoft.com/workshop/author/cpad/下载这个实用程序。还要确信你能找到 Site Builder Workshop 工具库中的其它工具。在 http://www.microsoft.com/workshop/default.asp 能找到关于这个工具的全部细节。在学习本章中其它例子时，最好先得到这个实用程序，因为，这些例子都假定你拥有 ActiveX Control Pad（从第 8 章开始，我们就讨论了这个实用程序，如果你没读第 8 章中“ActiveX 放在什么地方”这一节的话，现在去读一下吧）。

程序列表 9.1 是一个相当简单的 JavaScript（与 HTML 有关联的）脚本，用于显示一份订货单，并自动计算用户应付费用的总额。仔细看一下程序列表，有一两个标记以前还没见过，以及一些虽然你已经见过、但现在才第一次使用的标记。图 9.3 则显示了这张订货单，请注意表格这时是全部都能够看到的。这个特定的窗体也不包括 Submit（提交）按钮。本例的重要信息是如何在本地处理窗体中包含的数据。

程序列表 9.1

```
<HTML>
```

```
<HEAD>
<TITLE>Typical Entry Form</TITLE>

<SCRIPT LANGUAGE="JavaScript">
<!--
function CalculateItem1(nAmount)
{
      window.document.DataEntry.Item1Ex.value = nAmount * window.document.
          DataEntry.Item1.value
      CalculateTotals()
}

function CalculateItem2(nAmount)
{
      window.document.DataEntry.Item2Ex.value = nAmount * window. Document.
          DataEntry.Item2.value
       CalculateTotals()
}
function CalculateItem3(nAmount)
{
      window.document.DataEntry.Item3Ex.value = nAmount * window.document.
```

```
            DataEntry.Item3.value
        CalculateTotals()
}

function CalculateTotals()
{
        with (window.document.DataEntry)
        {
        //Calculate the item subtotals.
        nTotalItems = parseInt(Item1.value) + parseInt(Item2.value) + parseInt(Item3.value)
        nTotalAmount = parseFloat(Item1Ex.value) + parseFloat(Item2Ex.value)+
parseFloat(Item3
Ex.value)

        // Calute the total amounts.
        nHandling = nTotalItems * 5
        nTax = nTotalAmount * .05
        nTotal = parseFloat(nTotalAmount) + parseFloat(nHandling) + parseFloat(nTax)

        // Store the data.
        Handling.value = nHandling
```

```
        Tax.value = nTax
        Total.value = nTotal
        }
}
//-->
</SCRIPT>

</HEAD>
<BODY>
<!加入一个标题 . - >
<CENTER><H1>Sam's Special Order Form</H1><CENTER>
<H3><MARQUEE WIDTH=50% LOOP=INFINITE>
        Get It While It's Hot! Sam's is known the world  over  for  really  HOT
items! Sams will not be held responsible for stolen items, but  then  you're
not really worried about that if you're shopping here.
</MARQUEE></H3>

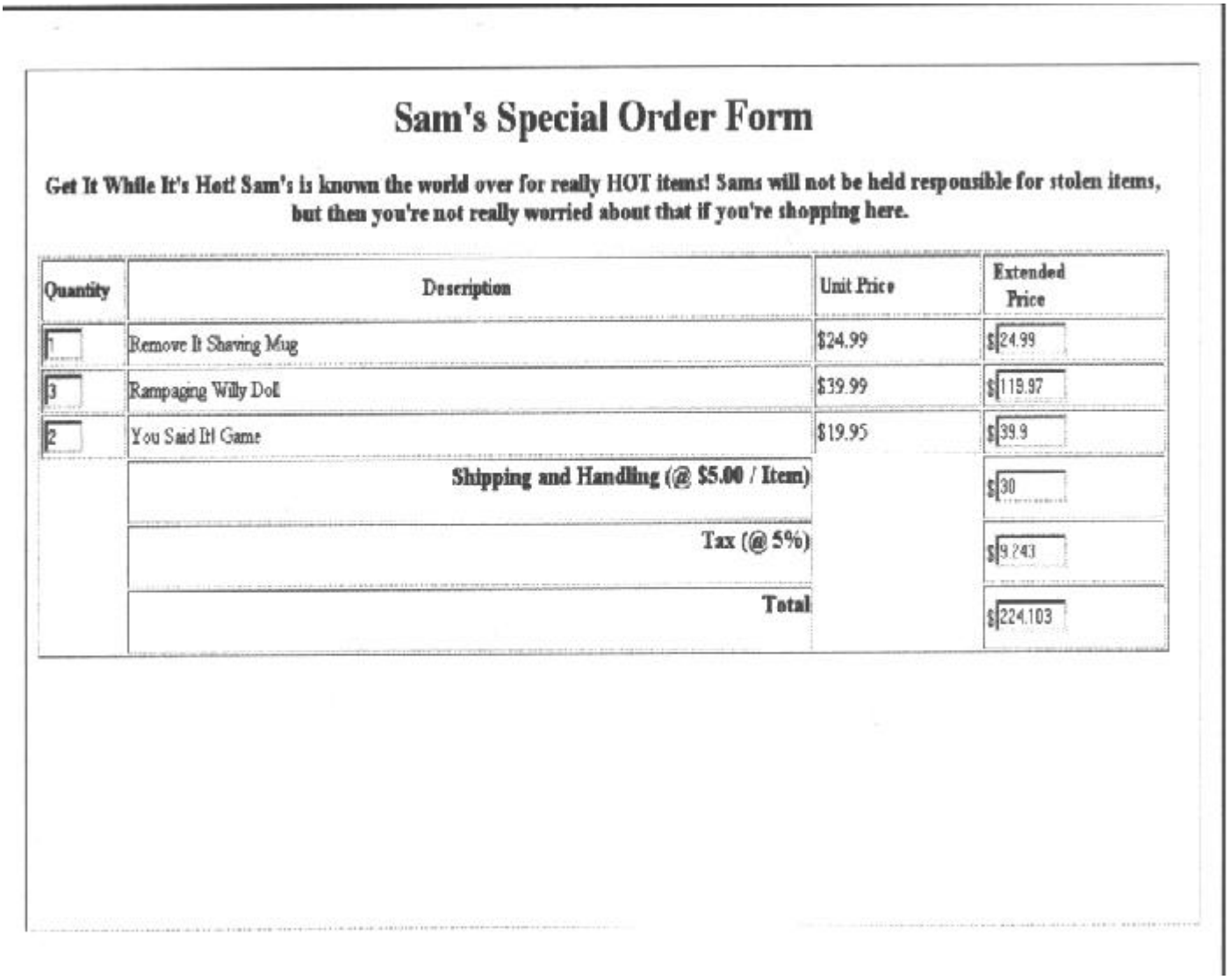
<!创建一个窗体 . - >
<FORM NAME="DataEntry">
<!-把订货项目表格放入窗体，表格包括四列：quantity(数量), description(说明),
<!-unit price(单价)和 extended price(合计)，用户只需在 quantity(数量)栏中->
```

<!-**打入数量即可**->

```
<TABLE WIDTH=100% BORDER=1>
<TR>
    <TH WIDTH=30> Quantity</TH>
    <TH>Description</TH>
    <TH WIDTH=75> Unit Price</TH>
    <TH WIDTH=75> Extened Price</TH>
<TR>
    <TD><INPUT  TYPE= "text"  NAME= "Item1"  VALUE= "0"  SIZE= 3
        ONCHANGE=CalculateItem1(24.99)></TD>
    <TD>Remove It Shaving Mug</TD>
    <TD>$24.99</TD>
    <TD>$<INPUT TYPE="text" NAME="Item1Ex" VALUE="0" SIZE=7></TD>
<TR>
    <TD><INPUT  TYPE= "text"  NAME= "Item2"  VALUE= "0"  SIZE= 3
        ONCHANGE=CalculateItem2(39.99)></TD>
    <TD>Rampaging Willy Doll</TD>
    <TD>$39.99</TD>
    <TD>$<INPUT TYPE="text" NAME="Item2Ex" VALUE="0" SIZE=7></TD>
<TR>
    <TD><INPUT  TYPE= "text"  NAME= "Item3"  VALUE= "0"  SIZE= 3
```

```
            ONCHANGE=CalculateItem3(19.95)></TD>
        <TD>You Said It! Game</TD>
        <TD>$19.95</TD>
        <TD>$<INPUT TYPE="text" NAME="Item3Ex" VALUE="0" SIZE=7></TD>
<TR>
        <TD><INPUT    TYPE= "hidden"></TD>
        <TD ALIGN=Right><H3>Shipping and Handling (@ $5.00/Item)</H3></TD>
        <TD><INPUT    TYPE= "hidden"></TD>
        <TD>$<INPUT TYPE="text" NAME="Handling" VALUE="0" SIZE=7></TD>
<TR>
        <TD><INPUT    TYPE= "hidden"></TD>
        <TD ALIGN=Right><H3>Tax (@ $5%)</H3></TD>
        <TD><INPUT    TYPE= "hidden"></TD>
        <TD>$<INPUT TYPE="text" NAME="Tax" VALUE="0" SIZE=7></TD>
<TR>
        <TD><INPUT    TYPE= "hidden"></TD>
        <TD ALIGN=Right><STRONG><H3> Total</H3></STRONG></TD>
        <TD><INPUT    TYPE= "hidden"></TD>
        <TD>$<INPUT TYPE="text" NAME="Total" VALUE="0" SIZE=7></TD>
</TABLE>
</FORM>
```

```
</BODY>
</HTML>
```



Sam's Special Order Form

Get It While It's Hot! Sam's is known the world over for really HOT items! Sams will not be held responsible for stolen items, but then you're not really worried about that if you're shopping here.

Quantity	Description	Unit Price	Extended Price
1	Remove It Shaving Mug	$24.99	$24.99
3	Rampaging Willy Doll	$39.99	$119.97
2	You Said It! Game	$19.95	$39.9
	Shipping and Handling (@ $5.00 / Item)		$30
	Tax (@ 5%)		$9.243
	Total		$224.103

图 9.3　Sam **的订货单窗体：自动计算出用户全部订货的费用总额**

让我们从讨论本例中的 HTML 代码开始。首先，它含有一个新标记<MARQUEE>，用于显示滚动的文本。用这一标记还能滚动图像——绝大多数程序员都这样做。这一标记的文本处理能力受到一些限制，加到<MARQUEE>标记中的“花体”文本不在屏幕上出现。开始时文本什么样，则文本一直保持这个样。这意味着在实际显示字幕之前，先要做些格式化（这也正是许多程序员宁可使用图像的原因）。

警告 <MARQUEE>标记不能在 Netscape Navigator 浏览器中工作。如果对标记进行的格式化正确，那么，在 Netscape Navigator 浏览器中看到的是如图 9.3 中所示的居中化了的字幕文本。使用<MARQUEE>标记时总是要使用 LOOP 属性，因为如果不使用 LOOP 属性，支持<MARQUEE>的有些浏览器就会做出奇怪的反应。你第一次实施这个特定标记时，注意收集用户反馈是非常重要的，这是由于在旧式的浏览器上，它非常容易引起麻烦。进行一些小改动，也许就能使你的站点上使用的这个标记在确实支持这个标记的浏览器中工作。

后面一段 HTML 代码是如何使用表格的范例。这里用表格创建了订货单。请注意几个部分都用了 WIDTH 属性，使用这一属性的方式有两种：要么给出占用整个屏幕的百分比，要么给出占用实际屏幕的具体数量。如果使用百分比，当用户改变显示的大小时，占用空间仍保持这一比例。还要注意，四个表头标题中的三个都用了 WIDTH 属性。Description（说明）标题则占用其它三项剩余的那一部分。可以发现，无论用户屏幕的分辨率是多少，也不管他们怎样改变浏览器的窗口大小，这种非常简便的方法都能向用户提供一个具有吸引力的窗

体。

<INPUT>标记中也有几个变化。代码中使用了 SIZE 属性，可以保持输入的文本有具体的大小。还可以看到<INPUT>标记的新类型。“隐藏”控件能够跳过表格中那些凸起部分。再把图 9.3 看一遍，就会注意到。在计算总额时，隐藏控件，通过恢复凸起的外观来保持 Quantity 数量域。

现在来看一下 JavaScript。我们把所有函数都放进了一个 HTML 注释中。理由非常简单，那就是防止旧式的浏览器读窗体时发生问题。还应注意到，HTML 注释标记的第二部分是//-->。如果不用双斜线（双斜线是 JavaScript 和 Visual C++ 中的注释），那么 Netscape Navigator 的 JavaScript 解释程序就试图对这一行做些事——好象它是一行代码；而 Internet Explorer 忽略这一注释行，它意味着能够简单地结束像-->这样的行。像注释标记这样的小事也能产生很大差别：你的代码是仅能在一种浏览器上工作，还是可以在多个浏览器上工作，理解这一点非常重要。

前三个函数相当简单。每一个都是监督着 Quantity 行元素的变化。发生一个变化后，就计算该项订货的合计金额，然后调用函数来计算订单中底部的总计栏。把三个计算组合到一个函数中也是可能的，这样做在一定程度上使程序更快，能减少代码的复杂度，而且也不增加函数调用开销。

可移植性 你会发现，Netscape Navigator 的 JavaScript 版本比 Internet Explorer 所提供的 JavaScript 版本要严格得多。你必须重视浏览器间存在的两个极重要的差别。前面例子的 JavaScript 代码（程序列表 9.1 中 CalculateItem1() 函数）中有一行：

Window.DataEntry.Item1Ex.Value=nAmount * Window. DataEntry. Item1. Value，它在 Internet Explorer 上工作得很好。如果看一下程序列表 9.1 中的同一行，就会发现两个差别：首先，代码的这个版本不区分大小写。看一下 Window 对象——当你在 Activex Control Pad 的脚本向导中查看时它是小写。另一方面，Netscape 却区分大小写，你绝对要打入小写的“window”才能使脚本工作。还应注意到，这时的 DataEntry 直接跟在 Window 之后，即 Internet Explorer 不要求使用完全限定的对象路径，而 Netscape Navigator 却要求使用完全限定的对象路径。到达 DataEntry 窗体的对象路径的完全限定版本应该是 window.document.DataEntry。在写出代码时，一定要牢记这些差别，否则，就要为找出脚本中的莫名其妙的错误而花费一个星期天了。

有必要对 CalculateTotals()函数看仔细些，因为在使用 JavaScript 窗体时，用了几个开始时你也许未注意到的技巧。使用 With() 函数，能够减少键入代码的数量：对于“window.document.DataEntry”，使用了这个函数后，只须打一遍即可，而不必每次用到时都打一遍。

由于 JavaScript 是一种松散类型的语言，本节开始时它直接保持变量实际类型的处理方式，也许会使你感到惊奇。事实上，在有些地方，它不能很好地完成任务，这时就需要做一些额外的工作。看一下第一组计算，可以注意到三个对象的值都被放到 ParseInt()函数调用之中，这个调用告诉 JavaScript 要把串转换成数值。如果不用这个函数，那么就是把三个串联结成一个长串，而不是按

要求将它们加起来。下一行也是这样的问题，这时，因为是三个带有美元符号的值，所以使用了 Parsefloat()函数。这样做虽然不是最好的方案，但它确实有效。

最后应该注意到，CalculateTotal() 函数使用了许多中间变量。中间变量帮助 JavaScript 直接保持变量的类型，并且使得代码可读性增强。如果担心中间变量使用的内存太大，少用一两个中间变量也是完全可以的。

技巧 在 Activex Control Pad 中，如果打算使用 JavaScript 语言取代 VBScript 语言，一定要用 Tools|Options|Script 命令进行。当前版本倾向于在每个会话完成后，就丢失这一设置，把 Script Wizard 所作变化逆转过来，需要用很多时间。

9.4 使用 ActiveX 控件

信不信由你，将脚本、HTML 和 ActiveX 控件组合起来使用，实际地创建一些令人震撼的 Web 页，你已万事俱备。现在需要做的唯一一件事，就是把你所学的全部知识装配起来，再附加一些信息就足够了。

和前面小节中一样，我们来看一个使用 JavaScript 的例子。差别是，现在我们把一切东西跟对象装配在了一起，因为我们在本书余下部分要研究更复杂的问题，所以这里的例子的目的非常简单。非常重要的是从现在开始要把基本知识牢记在心。

本节中，我们会看到使用 ActiveX Control Pad 将工作变得更容易完成的一种方式（尽管开头看起来是更复杂了）。我们全部使用 ActiveX 控件（从现在开始不再是<INPUT>标记）来创建一个半功能性的窗体。本节中假定使用 ActiveX Control Pad，但是也包含了全部所需的源代码，这样做潜在地使你能够用其它方法创建它。要做的第一件事是新创建另一个 HTML 页和一个使用 ActiveX Control Pad 的 HTML 框架。New（新建）按钮允许创建文档。

保存空的 HTML 框架。使用 Edit（编辑）|Insert HTML Layout（插入 HTML 框架）命令将框架插入到 HTML 页中，再添加一个标题以解释这一文档是干什么用的。HTML 页这时就完成了，在这个例子中不用再做什么事了。程序列表 9.2 列出了这个 HTML 页的代码。

程序列表 9.2

```
<HTML>
<HEAD>
<TITLE>Using ActiveX Control with JavaScript and HTML Layout</TITLE>
</HEAD>
<BODY>

<!-加入标题->
<CENTER><H1>Using JavaScript with ActiveX Controls On an HTML Layout</H1>

<!-插入框架->
```

```
<OBJECT CLASSID="CLSID:812AE312-8B8E-11CF-93C8-00AA00C08FDF"
        ID="Layout1_alx" STYLE="LEFT:0;TOP:0">
        <PARAM NAME="ALXPATH" REF VALUE="Layout1.alx">
</OBJECT>

</BODY>
</HTML>
```

注释 这个例子的 Web 页仅适用于 Internet Explorer。需要使用类似 NCompass ScriptActive 的插件使这个页在 Netscape Navigator 上工作。本书后面我们讨论这个过程。现在，仅适用于 Internet Explorer 的示例让代码简单些，从而使你能够清楚地看到它们的工作原理。

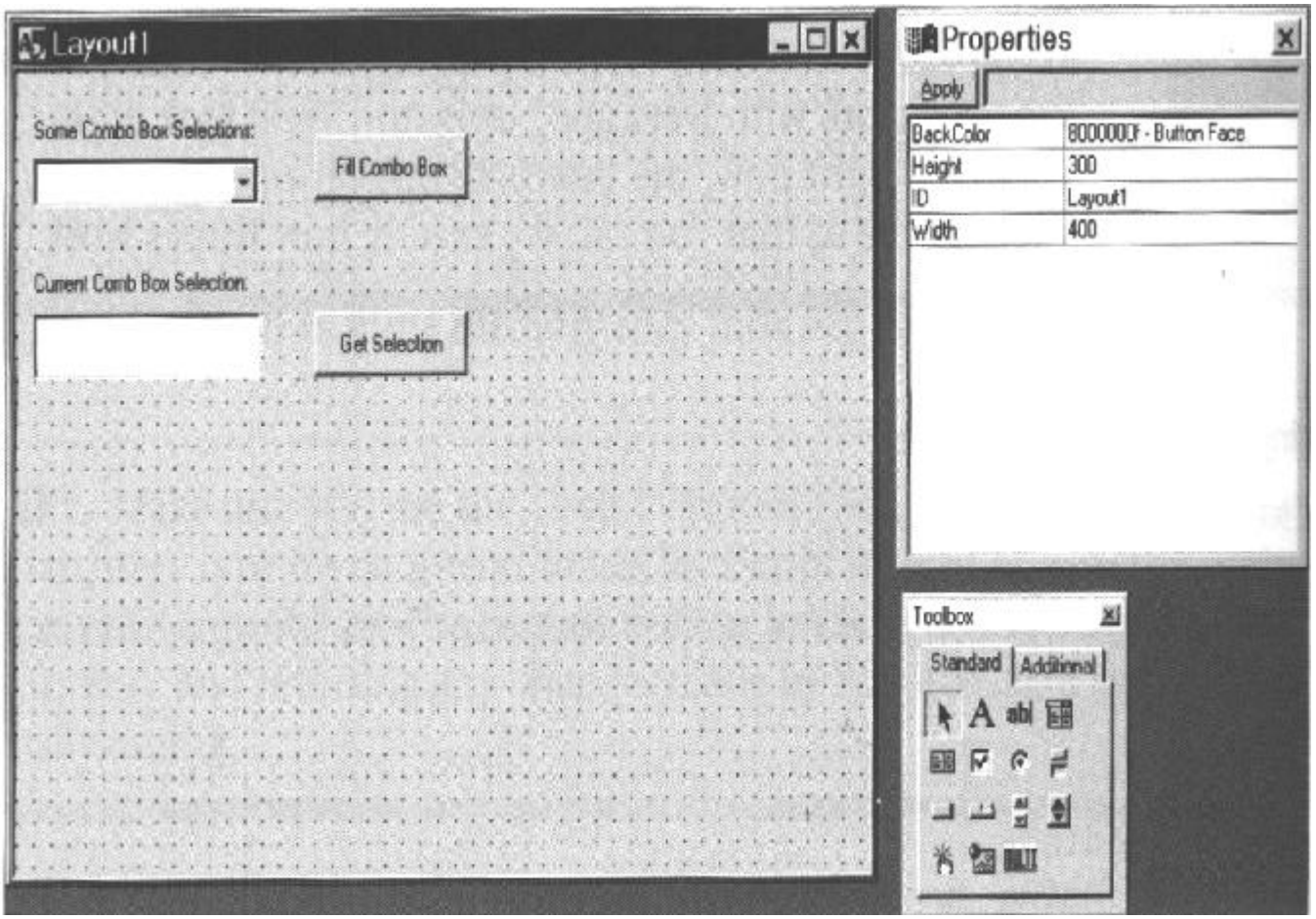


图 9.4　**使用** HTML **框架控件能够将** ActiveX **控件精确地定位到** Web **页上**

现在来干些容易干的事。首先看一下 HTML **框架。图** 9.4 **显示了本例中使用的一组** ActiveX **控件。你要完成的工作就是在** Toolbox**（工具箱）中单击一个控件，然后把它放到画布上。（如果喜欢的话，**ActiveX Control Pad **还允许使用拖放操作向画布添加控件。）控件放到画布后，可以把它拖曳到任何需要的位置，然后再改变其大小，使之与同一区域中的其它控件相匹配。功能框架上与图**9.4 **中所示的有些不一样，但该框架的设计目的是向你展示组合使用** ActiveX

控件和 JavaScript 的具体特色。请留意从 Toolbox 中取用的控件都是 ActiveX 控件。

该给这个示例增加一些功能了。和 VBScript 一样，JavaScript 也要求在向按钮或其它控件派发函数时，要执行一些特殊的函数名称格式化。还有一种容易的方法来产生函数名，这就是使用 Script Wizard（脚本向导）来产生函数名。使用 Tools （工具）|Script Wizard（脚本向导）命令，打开 Script Wizard（脚本向导），然后为一个按钮选择 Click（单击）事件，在 Script Wizard（脚本向导）对话框底部就会看到 List View（列表视图）和 Code View（代码视图）这两个单选按钮。单击 Code View 这个单选按钮，就会看到对话框底部发生变化，如图 9.5 所示。看一下对话框底部三分之一处的灰色条，注意到它清楚地说明了如何格式化你的 JavaScript 函数调用。若选择了 VBScript 代替 JavaScript 作为语言选择，那么，将显示出过程调用的 VBScript 版本。

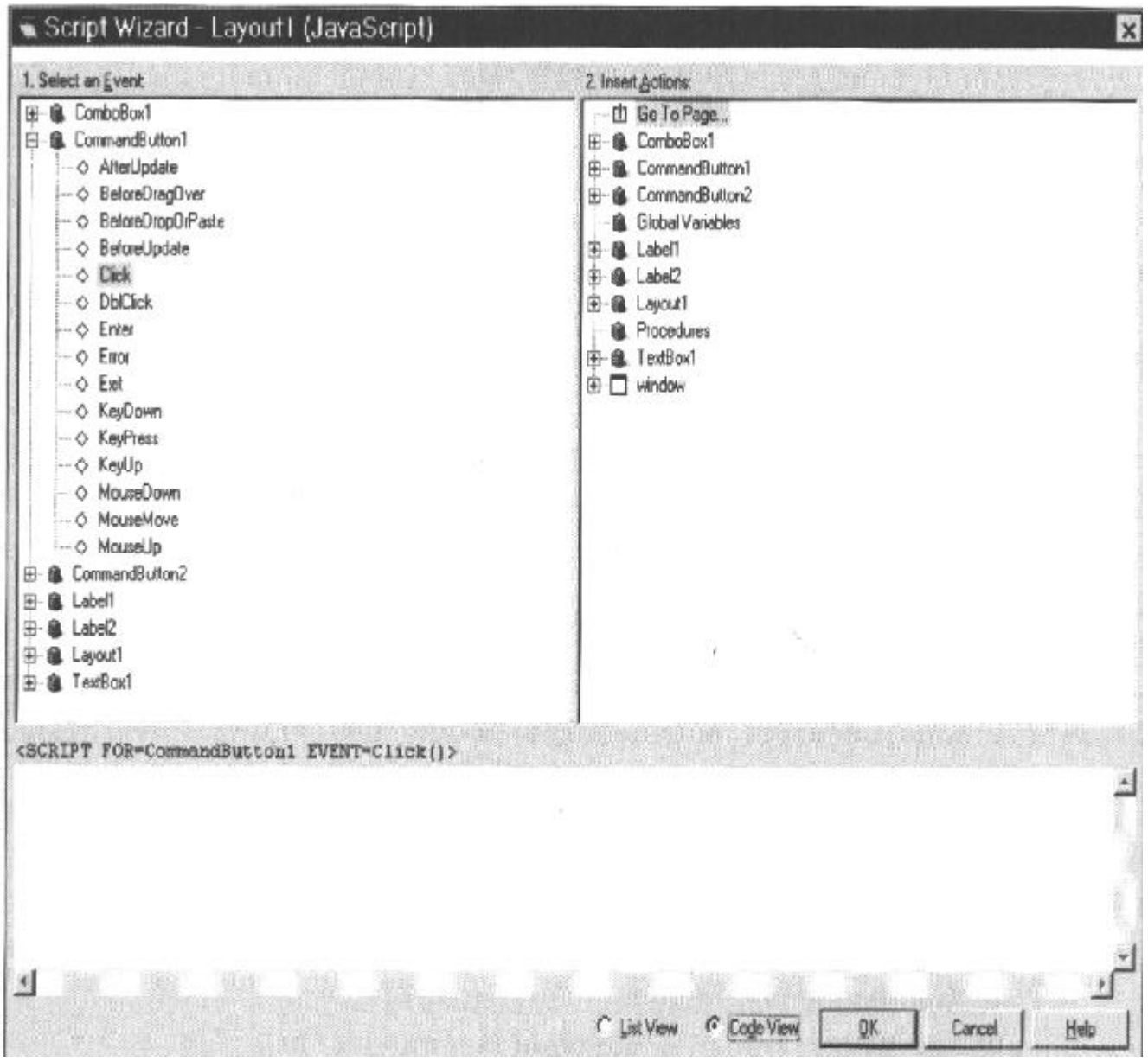


图 9.5　Script Wizard **帮助你处理一个对象所需的过程或函数的名称**

现在到了添加一些 JavaScript **使控件生效的时候了。关闭** Script Wizard **（脚本向导）（如果还没关闭的话），用右键单击框架，从关联菜单中选择** View Source

Code （查看源代码）。ActiveX Control Pad 可能要求你先保存框架，那么，按要求进行保存。然后就会看到一个 NotePad 或 WritePad（这取决于框架有多大）文本文件。不要破坏已存在的<OBJECT>标记，添加如程序列表 9.3 所示的代码（程序列表也列出了< OBJECT>标记。因为也许你会手工创建这个文件），本章最后的图 9.6 显示的是 HTML、 脚本和框架页组合起来在浏览器所看到的样子。

程序列表 9.3

```
<!-这个脚本用数据填充组合框->
<SCRIPT FOR="CommandButton1" EVENT="Click()">
        for (nCount = 1; nCount <ComboBox1.ListRows; nCount++)
                {
                ComboBox1.AddItem("Item " + nCount)
                }
        ComboBox1.ListIndex = 0;
</SCRIPT>

<!-这个脚本提取当前组合框的值->
<SCRIPT FOR="CommandButton2" EVENT="Click()">
        TextBox1.Text = ComboBox1.Text
</SCRIPT>
<!-这些对象都是机器创建的，不要修改->
```

```
<DIV ID="Layout1" STYLE="LAYOUT:FIXED;WIDTH:400pt;HEIGHT:300pt;">
    <OBJECT ID="Label1"
     CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0"
     CODEBASE="http://aux/controls"
STYLE="TOP:17pt;LEFT:8pt;WIDTH:124pt;HEIGHT:17pt;ZINDEX:0;">
      <PARAM NAME="Caption" VALUE="Some Combo Box Selections:">
      <PARAM NAME="Size" VALUE="4374;600">
      <PARAM NAME="FontCharSet" VALUE="0">
      <PARAM NAME="FontPitchAndFamily" VALUE="2">
      <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="ComboBox1"
     CLASSID="CLSID:8BD21D30-EC42-11CE-9E0D-00AA006002F3"
     CODEBASE="http://aux/controls"
STYLE="TOP:33pt;LEFT:8pt;WIDTH:107pt;HEIGHT:18pt;TABINDEX:1;ZINDEX:1;">
      <PARAM NAME="VariousPropertyBits" VALUE="746604571">
      <PARAM NAME="DisplayStyle" VALUE="3">
      <PARAM NAME="size" VALUE="3775;635">
      <PARAM NAME="MatchEntry" VALUE="1">
      <PARAM NAME="ShowDropButtonWhen" VALUE="2">
      <PARAM NAME="FontCharSet" VALUE="0">
```

```
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="CommandButton1"
      CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"
      CODEBASE="http://aux/controls"
STYLE="TOP:25pt;LEFT:140pt;WIDTH:72pt;HEIGHT:24pt;TABINDEX:2;ZINDEX:2;">
        <PARAM NAME="Caption" VALUE="Fill Combo Box">
        <PARAM NAME="Size" VALUE="2540;847">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="Label2"
      CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0"
STYLE="TOP:74pt;LEFT:8pt;WIDTH:107pt;HEIGHT:17pt;ZINDEX:3;">
        <PARAM NAME="Caption" VALUE="Current Combo Box Selection:">
        <PARAM NAME="Size" VALUE="3775;600">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
```

```
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="TextBox1"
     CLASSID="CLSID:8BD21D10-EC42-11CE-9E0D-00AA006002F3"
STYLE="TOP:91pt;LEFT:8pt;WIDTH:107pt;HEIGHT:25pt;TABINDEX:4;ZINDEX:4;">
        <PARAM NAME="VariousPropertyBits" VALUE="746604571">
        <PARAM NAME="Size" VALUE="3775;882">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="CommandButton2"
     CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"
STYLE="TOP:91pt;LEFT:140pt;WIDTH:72pt;HEIGHT:24pt;TABINDEX:5;ZINDEX:5;">
        <PARAM NAME="Caption" VALUE="Get Selection">
        <PARAM NAME="Size" VALUE="2540;847">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
```

</DIV>

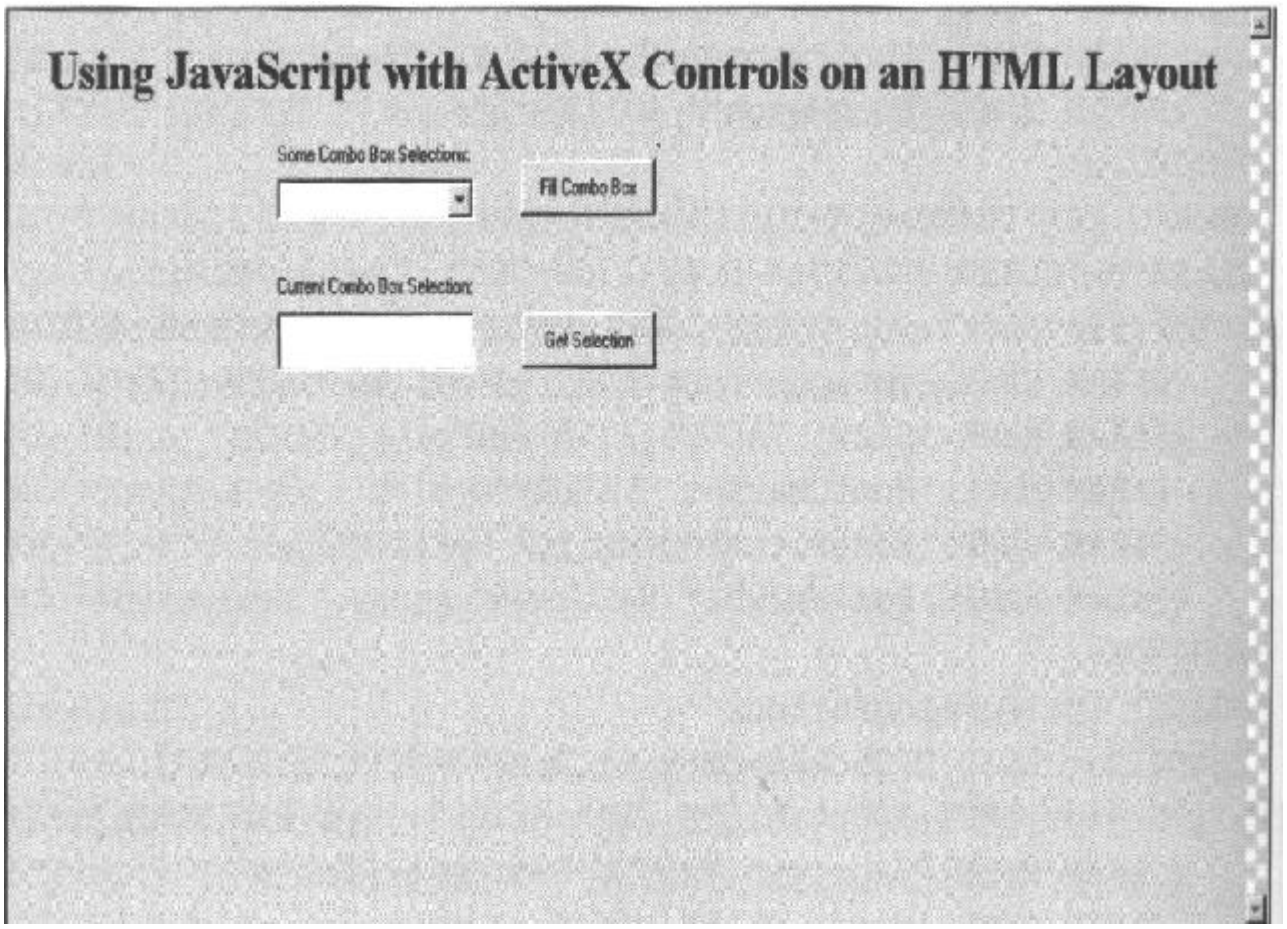


图 9.6　HTML **框架在浏览器中是不可见的，但使用它的效果是可见的**

程序列表 9.3 **中的两个脚本是很简单的，第一个脚本说明怎样填充一个组合框。真正要你做的，只不过是使用** AddItem（**添加项**）**方法告诉组合框你想添加些什么。填充了组合框后，使用** ListIndex（**列表索引**）**属性设置缺省的选择。如果不这样做，甚至在你填充之后用户还会继续看到一个空白的组合框。列表索引从** 0 **开始。**

使用 ActiveX 对象的问题之一是，不运行就不能做一些安装工作。例如，运行之前不能用值去填充组合框（用属性对话框试一试）。使用脚本语言操作 ActiveX 的整个想法是给你一种以前从未见到过的灵活选择。

存在多种方法能估计出用户做的选择是什么。若需要一个数值的选择数，那么可以从 ListIndex 属性中得到它。我们这儿需要一个串值，所以最好使用 Text 属性。这正是第二个脚本所完成的工作。它只不过是把 ComboBox1.Text 属性的值拷贝到 TextBox1.Text 属性中去而已。

第10章　创建ActiveX控件

某些人不愿意学习新东西，特别是学习看起来很难很复杂时更是这样。对许多人说来，OCX 就是这一类的。和在 DOS 环境下写设备驱动程序一样，写 OLE 控件也属于这一类。现在 Microsoft 出于某种考虑，已把 OCX 的新版本的名称改为 ActiveX，许多人会为不想试一试这种技术而找到更多的借口。

注　ActiveX 控件使用了 OCX 扩展名，就象他们设计用来代替的 OLE 控件一样。

因为你已经在阅读这一章了，所以我不能不假定你至少已对 ActiveX 控件背后的技术产生了较浓的兴趣。在本章中我们不会做出任何迷惑你们这些小伙伴的事。事实上，我创建的两个程序设计例子（非常有用）都非常普通。我们来看看如何创建一个新型按钮：on/off 开关。

选择这个例子的理由很容易理解，大多数人发现，与一些熟悉的事物打交道，可以缓解学习新东西的压力。另外，这个特定的编码例子实现起来极简单，只需用少量时间就能弄明白它是如何工作的。

注释　本章中的例子是用 Microsoft Visual C++6.0 开发的，使用 Microsoft Visual C++ 5.x 版或 4.x 版都不会有太多麻烦。用 Borland C++或比 Microsoft C++4.x 还老的版本编译写出的代码，不经修改是不能工

作的。创建能在每个产品下都能工作的万能例子，已经证明是行不通的。无论你决定使用哪一种程序设计语言产品，本章的例子都向你提供了关于开发 OCX 的有价值的提示或技巧点拨。

一旦写出了例子的第一个版本，应该学几种方法去测试这个新的 ActiveX 控件，看其是否完全安全（“安全”一词在这里意味着在一个产品环境中能使用这个控件，而不是对其安全漏洞进行核查，尽管你肯定想这样干）。有些人想一下子就把控件放入最近的 HTML 页或产品应用程序中，这实际上不是干事情的最好办法。通常在内部环境中(in-house) 测试三次，投入实际实现中时再测试一次。因为 Windows Scripting Host（WSH，窗口脚本主机）提供了使用 ActiveX 控件的另外机会，所以也要在这个环境中对它们进行测试。（如果 ActiveX 控件在 Internet 上不成功，你会发现可以限制它们用于公司内部网、WSH 附件或作为一个应用程序组件）。

注 一个 ActiveX 控件的完全测试，意味着对它能否在应用程序、本地 HTML 和远程 HTML 环境中工作。还要测试它在 WSH 环境下能否工作。

本章还要花费时间去讨论两个 API，它们是比照着 ActiveX 控件的参照物 OCX 开始定义 ActiveX 的，可以发现，绝大部分的 OCX 已经工作得很像是 ActiveX 控件了。阻碍你使用它们的唯一障碍是，供货商用什么方式许可使用它们。Internet 的公共特性使得使用你可能拥有的控件的当前版本更加困难。——大多数供货商允许把它们包括到你创建的应用程序中，但不能做为 Web 页的一部分。这些日子里，更多的关注是构造这些控件的方法，它们太大了，以至于不能在合理的时间内把它们下载。必须记住，用户不会仅仅是为了看到一个

花哨的屏幕按钮而乐意于下载一个很大的文件。

WEB 链接 附录 A 列出了也许会考虑添加到工具箱的第三方的 ActiveX 控件。在那里除了可以找到供货商外，还可以看一下由 NCompass 推出的 CaptiveX 控件。CaptiveX 包括 6 个控件，使你可以使用广告牌(billboard)和轻型招贴画(lightpanel)之类的方式显示信息。在 http://www.ncompasslabs.com/captivex/index.htm 可以获得 CaptiveX 控件的完全演示版。与许多演示站点不一样，如果安装了 ScriptActive 插件(可以从这个演示站点下载)，就可以使用 Netscape Navigator 在这个站点工作。

本章的最后一节，打算考察一下 RealAudio。我发现，看一下别人在做什么，至少与本章中我们钻研学习程序设计技术一样重要。RealAudio 提供一个控件，允许你在 Internet 上演奏音乐或其它种类的声音。事实上，可以发现这个控件的使用遍及 Internet，原因是它满足了你要知道的两条规则：第一，这个控件很小，用极少的下载时间就可完成下载任务；第二，它提供的是流型声音，这意味着用户可以得到即时反馈，实质上，用户不会注意下载整个声音文件要用多少时间，因为 RealAudio 控件只要得到文件的一小部分就可以开始演奏音乐（或其它声音信息）了。

10.1 理解ActiveX控件的一些背景知识

在真正开始创建 ActiveX 控件之前，需考虑三个问题。第一个当然是 ActiveX 控件是什么，这个问题引起了不少混乱，所以对它下个定义是很重要的。请记住这是本书的定义，一旦你开始成为迎合 ActiveX 程序员的新闻组的常客时，就会发现各种其它答案。第二件事是，我们要讨论一下，一旦你创建了一个 ActiveX 控件，你能从它得到些什么，这个讨论中还要看用户期待些什么。最后，我们需要看一下 ActiveX 控件与使用过的其它控件有什么区别，最重要的是 OCX 和 ActiveX 控件有什么区别。下面每一节帮助你探讨刚才提到的一个问题。

什么是ActiveX

这里讲得是你发现的对 ActiveX 所作的最简单的定义。ActiveX 是 OCX 的高级形式（也许把它看作 OLE 的较简单形式更合适）。但是，这个简单定义不能表达 ActiveX 控件的实质，用户认为 ActiveX 就是 OLE 。对程序员来说，ActiveX 还是一套用于 Internet 的实能技术。它提供了过去没有的一种信息交换方法。

注释 Microsoft 目前正在开发一种新的 Internet 专用的组件技术，称为 COM+ 。尽管许多内部网使用了 ActiveX，而且 ActiveX 依然将用作应用程序组件，但 ActiveX 从未在 Internet 上广泛被采用过。ActiveX

在 Internet 上失败的理由有三条：组件大小，安全考虑以及缺乏对非 Microsoft 浏览器的兼容性。请谨记，你或许应该考虑把 ActiveX 控件作为公司内部使用的解决方案，而不是做为 Internet 的实现技术。

作为真正欣赏 ActiveX 的程序员，就不得不从程序员的角度去看一下 OLE，这就是意味着要看一下 OCX。从用户的角度看，OCX 所做的都是在两个应用程序（或操作系统与一个应用程序）之间交换数据。OCX 比数据交换的内涵要丰富得多。它们包括着一种称之为组件对象模型(COM)的概念。COM 是定义对象模块间标准二进制接口的规范。这个接口定义了函数调用方法、基于标准结构的数据传送技术以及几个标准的函数调用。使用 COM 意味着，你写应用程序模块（如 OCX）时使用何种语言都无关紧要，因为模块接口在二进制层次上是相同的。

注释 写作本书时，Microsoft 正在设计 WebView，它是 Internet Explorer 与 Windows 95 系统的 Explorer 之间的一种集成技术。这一新技术使得对 Web 站点的访问就像对 Explorer 中列出的驱动器或其它资源的访问一样容易。还可以看到你比较熟悉的 URL 已被普通英文名称所代替。你可以在 Windows 98 中看到这一技术的实际运用。

COM 怎样对你写的应用程序产生影响呢?答案是相当复杂的，原因在于使用 COM 的方式众多，倒不是因为技术本身是多么复杂。当用户把一个图形图像对象放入你的应用程序控制的容器中时，你对这个对象知道些什么呢? 实际

上你只知道是谁首先创建了它。知道这一信息就允许你在这个应用程序中调用许多服务，这些服务包括显示这个图形或允许用户编辑它。实际中，你在做的是共享应用程序的代码。

程序员也能从使用 COM 中受益。当把一个 OCX 安装到你的程序设计环境中时，实际上你完成了什么呢?大多数情况下，你拥有了一个粘贴于窗体某处的一个新控件。你不必真正地知道控件的内部工作原理，唯一重要的因素是，控件能为你的应用程序做些什么，以及如何与它打交道。你可以使用标准接口来调用安装到你机器中的特定代码模块，这就是 COM。

ActiveX 是这种思想的扩充，仍然使用标准接口，但是，不再简单地从本地机器环境或 LAN/WAN 网的常设连接上调用代码，而是从 Internet 调用它。另外，这一新的代码能采用 applets（包含内容的小程序）或微小应用程序的形式。

ActiveX 将为你做些什么

OCX 能为桌面系统做些什么，ActiveX 就能为 Internet 做同样的事。但是，你将在以前从没想到的地方发现 ActiveX 控件。例如，NetManage,Inc 公司计划创建一个新的电子邮件客户机，称之为 Z-Mail Pro。这一软件包支持 ActiveX 技术的方式是，允许用户直接地在消息查看窗口中交换、创建以及查看 HTML 文档。这就意味着用户拥有了创建动态 Web 页的能力，这种能力正是你今天不得不实际完成的任务。

远程连接也会从 ActiveX 的使用中受益。例如，Proginet Corporation 目前正在开发把主机(Mainframe)数据带到桌面系统去的 ActveX 技术。它的 Fusion

FTMS（文件传输管理系统）可以使用支持 OLE 容器的任何开发语言，比如 Delphi，Visual C++以及 PowerBuilder。从本质上看，你将把一个 ActiveX 控件放入一个窗体，定义到哪里去找数据，然后依靠控件进行连接。Internet 上的远程访问不再要求用户去钻过马戏团中的大铁圈了（即不用在连接中跳来跳去了）。主机上一个特殊的传输服务器通过自动化地处理所有传输请求而完成打包。操作员在客户能访问公司站点之前再也不用人工地为站点下载一个必需的文件了。

甚至于 Microsoft Exchange 也会从 ActiveX 受益。Wang Laboratories,Inc. 和其它公司正在创建将 Exchange 和 ActiveX 混合在一起的新的附件。Wang 的产品是客户机/服务器图像附件，它允许用户扫描、查看、注释、操作或打印图形图像， 而不用考虑这些图像位于何处。正是这个产品，还包括了一个层次存贮管理的 ActiveX 控件。将这两种技术组合在一起，使得在大公司中使用和访问图形变得容易了。它们还使得寻找一个所需的图形变得容易了，这最终导致公司节约了存贮空间。

Microsoft 自己也发布了许许多多的 ActiveX 控件。其中有些可以从 Microsoft 的 Internet 站点(http://www. microsoft. com) 免费下载，这些新的控件有用于 PowerPoint 的 Animation Player 以及用于 Access 和 Schedule+的 Internet Assistant（Internet 助手）。Access 的 Internet 助手能为要上载的数据库表格拍出快照作为静态图像，每当用户访问该页时，快照都会自动更新。Schedule+ 的 Internet 助手允许把日程表信息加载到 Web 页，因为每当用户访问站点时都会自动更新这些数据，所以再也不用担心在家上班的雇员会缺席会议了。最后，

PowerPoint 的动画放映机则允许你在任何兼容 ActiveX 的浏览器中演示 PowetPoint 的演示文稿。

最后，如果认为 ActiveX 无助于安全性，那还要再推敲一下。这些日子里，许多的新的防火墙和证书策略正在轮番出现，其中一个是 Net2000 。Net2000 是由许多 API 构成的，允许开发人员把 NetWare 内核服务（包括目录、 安全和特许证书）放入自己的应用程序中。你可以在内部网中通过 ActiveX 控件来选择这个 API。这种做法对用户和开发人员有什么帮助呢? 这意味着，在适当的程序设计结构下，网络管理员能够追踪整个网络中的特许使用，这种追踪甚至于跨越 Internet 连接。当更多的人开始在家中而不是在办公室中进行计算工作时，这就变成了一个非常重要的课题。

ActiveX 与 OCX 控件

就绝大多数情况来说，ActiveX 控件和 OCX 控件是完全可以互换的。可以看到与 Internet 风马牛不相及的 ActiveX 控件广告，仔细一看，就会发现，在不久以前，这些控件大概还出现在 OCX 控件列表中。当然，你要仔细观察这些控件。即使 ActiveX 控件与 OCX 控件确实共享了同一继承，ActiveX 控件与 OCX 控件也不完全是一回事。请记住，ActiveX 控件是能够在 Internet 上工作的。

Internet 对程序设计环境带来了一些特殊的挑战。你不再能够奢求高速装入，OCX 在因特网环境中使用时，其规模成了一个严重的问题，下载一个 60KB 的 OCX 就对用户的耐心带来了考验，而试图下载一个 200KB 的 OCX 大概会导致用户中断下载。而 ActiveX 控件是 OCX 的小型版本。

注 OCX 和 ActiveX 控件的一个主要差别是 ActiveX 控件通常要小一些（轻一些）。

ActiveX 控件也在受到各种机器专用需求的牵制。在机器上安装 OCX 时，安装程序能对机器进行检测并作出所需的许可。对 ActiveX 控件却不能这样要求。在这里你不能对客户机作任何假定，客户机可以是新的奔腾，也许是昨日的 80386。（如果你的 ActiveX 控件确存在某种平台限制，那么，要么设法解决它，要么肯定每个使用它的人都清楚这一限制。）

你还需要应付 OCX 程序员从未想到的情况。比如，如果浏览器根本不支持 ActiveX 时会发生什么?目前处理这一问题的方法是，浏览器对那些不知作如何处理的 HTML 标记只是简单地忽略掉。在这种情况下很容易应付一个与 ActiveX 不兼容的浏览器——只需打出一条消息，让用户知道他们的浏览器不能在当前页工作并指导用户换一个浏览器就可以了。

基于 MFC 与基于 ATL 的控件

Visual C++为了说明它比其它程序设计语言有完善的开发环境，它包容了两种创建 ActiveX 控件的方法：MFC 和 ATL（Active 模板库）。但是，这一灵活性也带来了其它开发人员没遇到的一些问题。比如，怎样确定创建哪一种类型的控件? 一些开发人员通过在他创建的所有控件中使用同一种技术来解决这个问题，但这样做本身就表明，你还没有真正地钻研并利用 Visual C++的全部潜力。

注 用 Visual C++创建 ActiveX 控件有两种方法：ATL 和 MFC。

确实没有任何理由能肯定地说，在给定的情况下，创建控件使用哪种方法

会更好。真正需要做的是弄清楚：你期待控件做些什么，你打算对什么进行开发从而获得所需功能，以及你的专业知识水平。显然，有些情况下，创建控件用这种方法比另一种方法更好，因为，两种方法确实各有利弊。当把一个 ATL ActiveX 控件与一个 MFC ActiveX 控件作对比时，需要考虑一些因素，为了给你一些对这个问题的基本看法，请读完下面列出的各条注意事项，从这里会发现一些对你有用的基本思路，从而帮助你选择一种最好的方案。

开发速度　使用 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）是创建控件的最快方法。这个向导将大部分接口细节管理起来，使你最终得到控件逻辑的一个轮廓。事实上，使用 ATL 方法创建控件通常要使开发人员用两倍（于 MFC 方法）的时间。显然，这还取决于诸如控件复杂度以及程序设计经验之类的因素。

维护　MFC 和 ATL 共有的特色是使得用它们产生的代码，比那些从头开始写出的代码，维护起来要容易得多。但是，当对一个原来的控件作一些与维护有关的变化（比如找出小毛病后修改掉它）时，就要考虑一下了。ATL 控件通常包含不多的由向导生成的“锅炉钢板型”（即难懂的）代码。因为对自己写出的代码会更加熟悉，因而改变起来就更快更容易。另一方面，Microsoft 维护着你使用的 MFC 中所有控件，这就是说，许多小毛病的改正和其它种类的更新，只要简便地再把代码编译一次，就可以自动地完成。在这种情况下作何种选择，确实难以回答，因为没有办法能确切地确定在将来你需要对你的控件作什么样的维护变化。

控件大小　如果要创建最小的 ActiveX 控件，那么采取 ATL 方案最好。ATL 把控件的各方面控制权都留给你，使得你可以方便地手工调节控件的每个

元素，而不至于陷于 MFC 专用代码的泥潭。基于 MFC 的控件不仅规模大，而且在用户使用这一控件之前还必须下载 MFC 库，这个库也有相当多的代码。

学习难度　由于创建 ATL 控件时不得不考虑更多的事情（比如接口），所以创建起来比 MFC 控件就更困难。多数情况下，先用 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）创建几个控件，从中学习创建控件逻辑的规则，这样做是值得的。

技巧　如果你用惯了 Visual C++5.0 的 ATL，那么就尝试一下 6.0 版中新的 ATL COM AppWizard（ATL COM 应用程序生成向导）。新的 ATL COM AppWizard（ATL COM 应用程序生成向导）帮助你以比过去手工技术快得多的速度创建控件。但是，使用 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）创建控件仍然要比使用 ATL COM AppWizard（ATL COM 应用程序生成向导）快得多。

- 兼容性　从定义可以看出，基于 MFC 的 ActiveX 控件要求客户将 MFC 库安装到他们的机器上。但是，有好多种版本的 MFC 库在使用，而它们并不都是相互兼容的。如果用户下载了你的控件及其相关的库，但不能使用某些重要的应用程序，因为新库与它们的应用程序不兼容，这时该怎么办呢? 因为 MFC 库存贮于 SYSTEM 目录下，客户机仅能拥有一个版本，到那里找到 MFC 库就能解决兼容性问题了。
- 使用的难易程度　MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）对你写控件代码的能力不作多少要求，所以它把它所拥有的一

切都放进了控件之中。这意味着你最终可以得到十分丰富的接口，但其中有的对你来说或许用不着。这一功能性浪费，导致了控件规模的膨胀，并且使得它难于使用。

- 代码修改的难易程度　开始时，创建基于 MFC 的控件是非常容易的。因为向导把许多代码都添加好了，所以应用程序开发进程很快，你只需关注把你的控件做得出色的细节。如果创建的控件投入使用后不需作什么修改，选择 MFC 显然是上策。但是，如果决定要更新控件时该怎么办？你的源代码文件包括了大量代码，这些代码不是由程度员写出来的，需要用额外的时间去研究、理解它们。因此，如果计划经常会对代码作修改，那么选择 ATL 将是上策，因为，当控件中由程序员生成的代码出了什么事时，程序员处理起来就容易多了。

10.2　一个基于 MFC 的基本按钮编程示例

在我的机器上，我想不出有哪一个程序不用到一两个按钮。事实上，按钮是为数不多的、可以说是每个应用程序都要用到的 Windows 控件之一。即使是在一个简单对话框中显示信息的应用程序中，通常也要在用完对话框后关闭它时使用 OK 按钮。只要说按钮（或某些程序设计环境中所说的命令按钮）是使你机器正常运转的一个控件就够了。它还是绝对能为你提供全部所需特色，而且程序员改变最多的一个控件。

注　按钮是每个应用程序都使用的一个控件，也是对定制控件的潜力进行

讨论时最好的入口点。

程序员对一个控件的用途作了这么多的强调，但这并没有用去我多少时间，使我不能在本章的这一节中，确定用哪一种控件向你说明如何去对它进行修改。我还想增加一个新特色，你也许在按钮中并没有发现这一特色，而这个特色最终是每个人都必需的。正是为了说明下述特色，我才选择 On/Off 按钮作为本章中控件的基础——就好象在进行应用程序设计中每个人要打开/关闭某个东西一样。将这种特色内置于你使用的控件之中是非常好的。对于从 Visual C++中得到的缺省控件来说，我还增加了其它几个特色。尽管我的这个按钮也许不能将商业化程序中所用控件的花哨特色全部囊括，但它能帮你开始创建你自己的定制控件。

定义工程

从用户和程序员两个角度来说，本章前面讨论过开发 ActiveX 控件的一些规则。现在再从程序员角度深入探讨一下。在开始讨论第一个程序设计例子之前，我们快速地看一下，定义一个具体的 ActiveX 控件必须记住哪些规则。下列的建议可以加深你对创建 ActiveX 控件必须遵守的一些特殊约束的理解。

技巧 尽管使用能生成 OCX 的编译器就能创建 ActiveX 控件，但你会发现使用 Microsoft Visual C++（4.2 版以上）产品将会为你节约许多时间。你在本章还会看到，要使 OCX 像 ActiveX 控件那样正确地工作，哪怕在最好的环境中，也需要一些额外的步骤。使用旧版编译器意味

着你需要用 DOS 命令行工具来处理额外的步骤。Microsoft Visual C++ 编译器的 4.2 及 5.x 都把这些工具作为软件包的一部分，而且将它们的使用自动化了。

- **保持代码小型化** 经验告诉我们，应该保持 ActiveX 控件的体积在 40KB 以下。通常用户不想下载庞大的、仅能用于对页面图形作动画处理的控件。实在没办法时，把一个大的组件分成几个小的功能块来实施。

技巧 还可以像第 8 章讨论过的那样，把控件压缩进 CAB 文件也能够减少下载规模。

- **使用最少的持久数据** 一些 OCX 要求大量的持久数据来完成任务。例如，你可能会把一个电子表格控件粘贴到窗体上，而没想到它所包含的持久数据的数量。ActiveX 控件没有这般奢侈，因为你不能对客户机有太多的假设。持久数据不仅增加装入时间和内存需求，而且扩大了控件自身的规模。
- **将功能特色限制在最低限度** 众多的功能的确能为屏幕演示增光添彩。为本地机器编写 OCX 时，几个额外特色不会增加什么问题。事实上，如果不包括它们就会使多数程序员感到吃惊。但传输时间的确是 Internet 的一大问题。向控件中每增加一个不是真正必需的图形或其它特殊声音效果，都要增加装入时间，并且降低控件的价值。ActiveX 控件甚至不能假定一些特殊效果是否能够在客户机上正确工作（举例来说，客户机可能没有声卡）。
- **功能单一是关键** 建库的原始动机之一是，用容易访问的形式，存贮大量预编译过的函数调用。作为 OCX 的前辈，DLL 存在的理由正在于此。你会

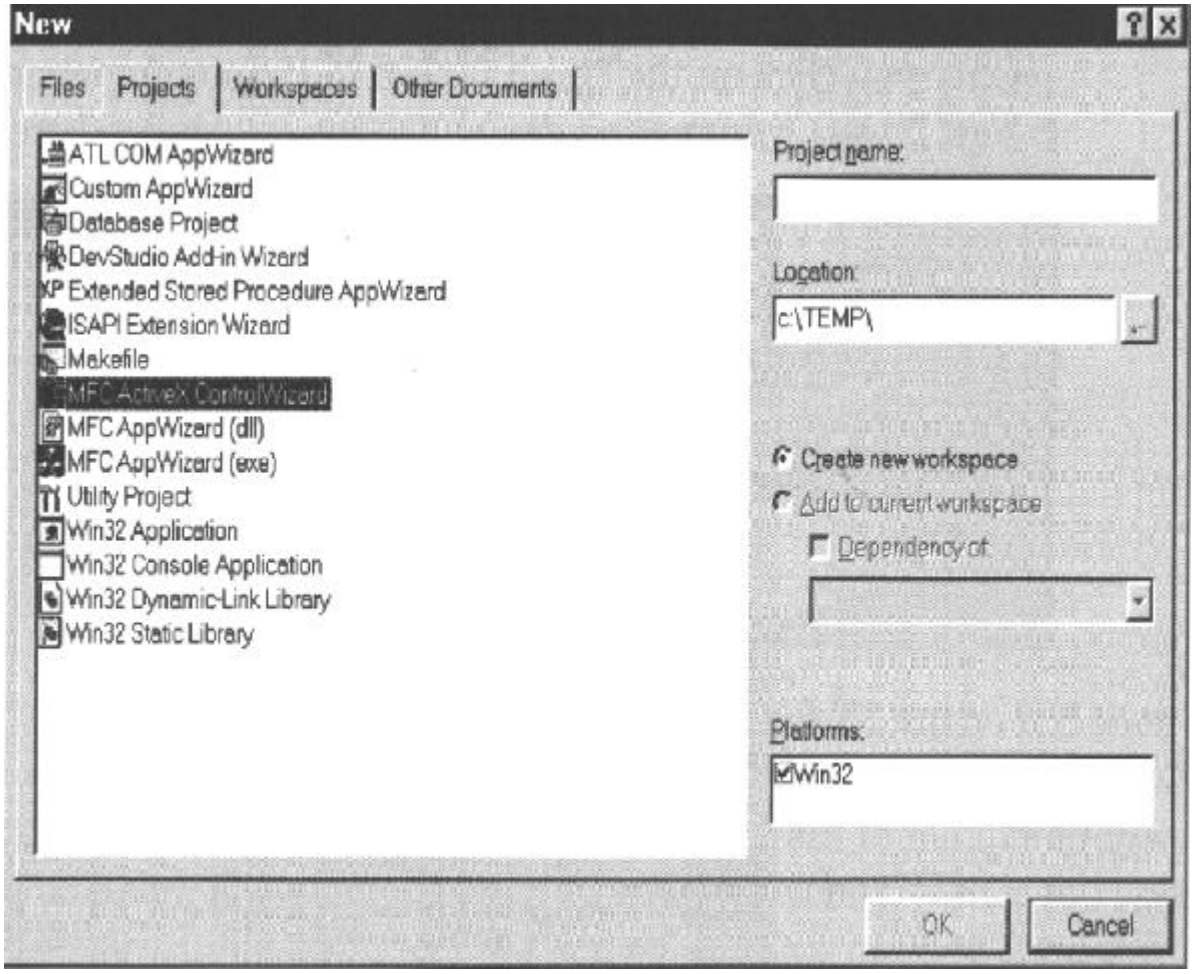
发现市场上的许多 OCX （如 DLL）都不止包含一个对象（如按钮）。事实上它们包含了一簇对象。这一策略对于桌面系统来说当然很不错，但到了 Internet 上就行不通了。要保证在一个控件中保持一个对象。遵循这样的原理去模块化控件，确保了用户不必下载他们绝对需要功能以外的东西。

注释 本章中还将对对象进行相当多的讨论。本章中每个对象都是 Windows 对象（或 COM 使用的特殊形式）。一些 C++程序员可能认为 Windows 对象就是 C++ 对象。因为 COM 使用一种特殊形式的 Windows 对象，这些程序员就认为，使用手边的 C++对象写 OCX 不会出问题。越过真理一步就是谬误。尽管可以用 C++对象创建 Windows 对象，但存在一些限制。本章篇幅不足以全面讨论 C++程序设计中对象的种种复杂性——这个题目，一些作者要用一章或两章的篇幅来作综述性讨论——但是，了解一下对于用 C++写 OCX 时能做些什么以及应做些什么、有哪些限制，都是很重要的。最好按照本章例子中的方法去写 OCX，这样才不会出问题。

测试、测试、再测试 仅仅在本地或仅仅在网络上测试 ActiveX 控件都是不够的。应该使用不同的连接，在多种情况下进行测试。本章最后会看到三个层次的本地测试以及一个层次的针对 Internet 的测试。可能你想再加进几个层次的测试，根本上说，你对一个 ActiveX 测试永远不嫌多。（在发布控件之前测试它是重要的，但在出问题后一定要去查出缺陷，所以要使 ActiveX 控件正常工作，维护精确的问题日志也很重要的。）

编写代码

现在到了看一下简单编码示例的时候了。本章使用 Microsoft Visual C++6.0（尽管你能够使用 4.2 以上版本），这在前面已提到过。让我们开始创建一个 C++新工程。但是，与你创建过的其它工程不一样，现在要用 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）来创建工作区。现在开始吧，使用 Flie（文件）|New（新建）命令，系统显示 New（新建）对话框，选择 Project Tab（工程页），则会看到如下所示的对话框。



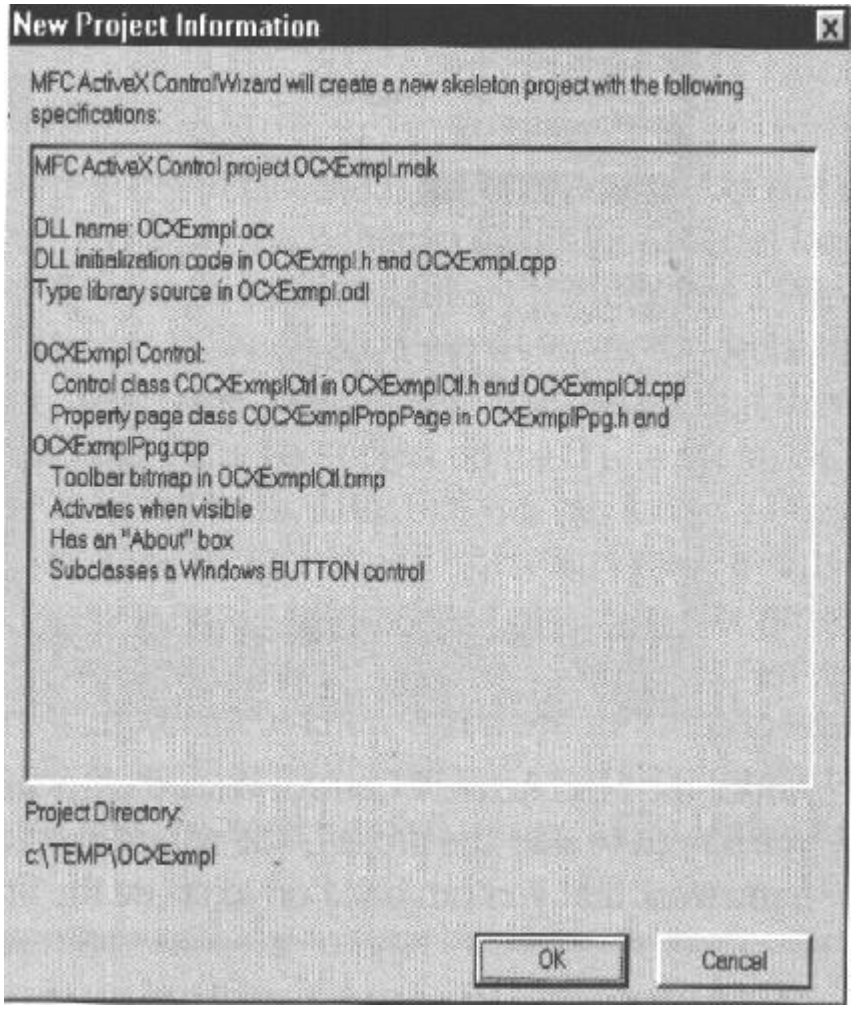
该对话框中的 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）就是开始创建本工程的那个选项。这个向导提供了一个 OCX 框架，通过建立这个框架来创建本例的最后版本。

注释 本例中我使用了新的 Microsoft Developer Studio（开发人员工作室）。看到的所有的屏幕图像都是 Windows 95 下的。如选择使用老的界面，你的屏幕图像就会与这儿的图像不一样。即便你也在使用 Developer Studio 接口，因为这一产品提供的配置选项可以修改，因此，仍可能产生一些小的差别。

工程开始时，在 Project Name（工程名）域中键入工程名。本例用 OCXExmpl 作为工程名。在工程列表框中单击 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导），然后单击 OK。Microsoft Visual C++自动选择了 Win32 选项，它还创建了一个工程目录。

接下来你看到的是 MFC ActiveX Control Wizard（MFC ActiveX 控件生成向导）屏幕的两个对话框。我采用了这两个对话框的缺省设置，但对第二个对话框的 subclass（子类）域作了选择，如果你也想创建像这个例子一样的控件，那么，在这个域中选择 BUTTON 类。否则，可以浏览可用类的列表框以确定用什么作为你的控件的基础。还要注意到 Visual C++允许创建自己的基类。

在第二个向导屏幕上单击 Finish（完成）按钮后，则会看到如下的 New Project Information（新工程信息）对话框。

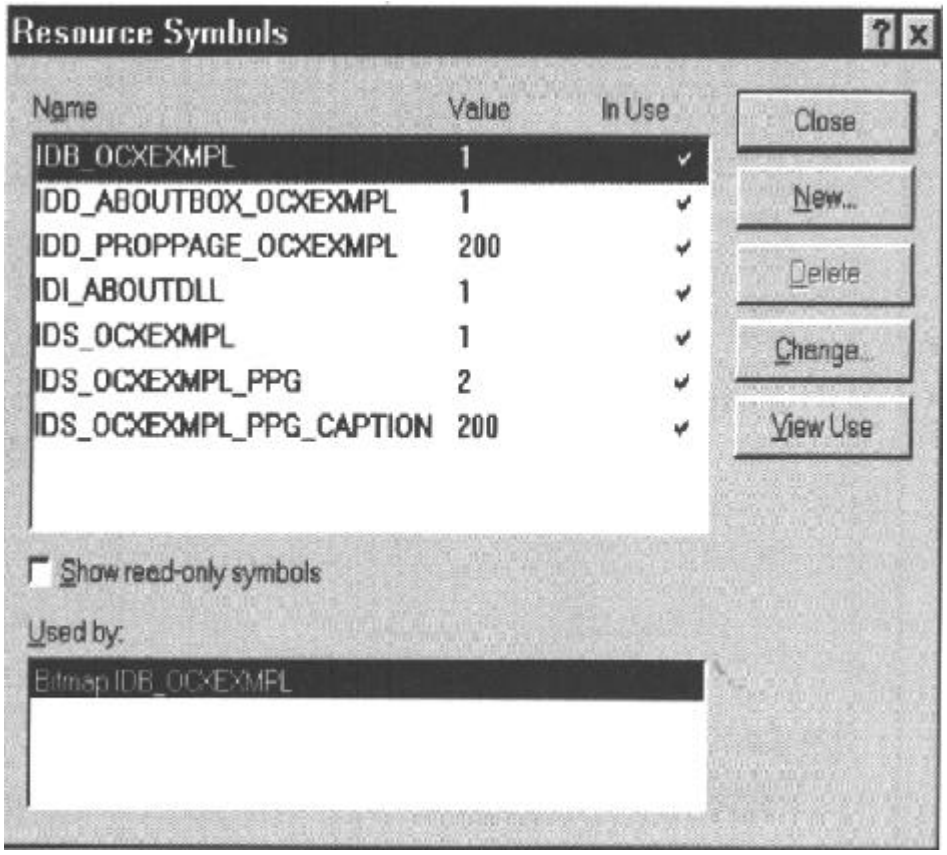


浏览一下所提供的功能列表，确认工程中包含了你所需要的一切内容。核对工程建立无误后，单击 OK，则启动了工程。Visual C++将使硬盘转动一会儿，然后就会看到工程框架。

修改缺省的 About 对话框

现在已有了一个框架，我们把它填充好。我总是先处理简单的事情（谁不是这样呢？）。首先修改 About 对话框。Visual C++ 自动为你创建这个对话框，

你需要做的事就是定制它。访问 About 框很容易，使用 View（视图）|Resource Symbols（资源符号）命令则显示出如下图所示的 Resource Symbols（资源符号）对话框。



选择 IDD_ABOUTBOX-OCXEXMPL 项，单击 View Use 按钮显示该对话框。图 10.1 显示了修改示例中 About 对话框的一种方法。也许你想把版权信息以及公司信息也包含到 About 对话框中。请留意 Microsoft 为对话框提供了多种工具，其中之一是定制控件按钮，你能够把另一个 OCX 贴到 About 对话框或其它你创建的对话框中。

技巧 用右键单击一个控件，然后从上下文相关菜单中选择 Properties（属性），于是 Properties（属性）对话框显示出来。通过改变 Properties（属性）对话框中 General（普通的）属性页上的 Caption（标题）属性，可以改变静态文本控件的文本。用右键单击 Visual C++的大部分对象，都会显示一个上下文相关菜单——大部分这样的菜单中都包含 Properties（属性）选项。用右键单击 Visual C++中某个对象的目的，就在于让你看到能够对这个特定对象做些什么。请记住，对象不仅仅指控件，它也可以是代码行、工具栏、甚至可以是各种窗口。

Visual C++的最新版本可能要求你对 About 对话框做一些过去不必做的事。你可能需要为 About 对话框创建一个类——Visual C++的老版本只是假定你会创建新类。双击 About 对话框，则看到 Adding a Class（添加类）对话框，如下图所示（若没看到该对话框，则说明已有一个类赋给了 About 对话框，你不必再往下继续做了）。

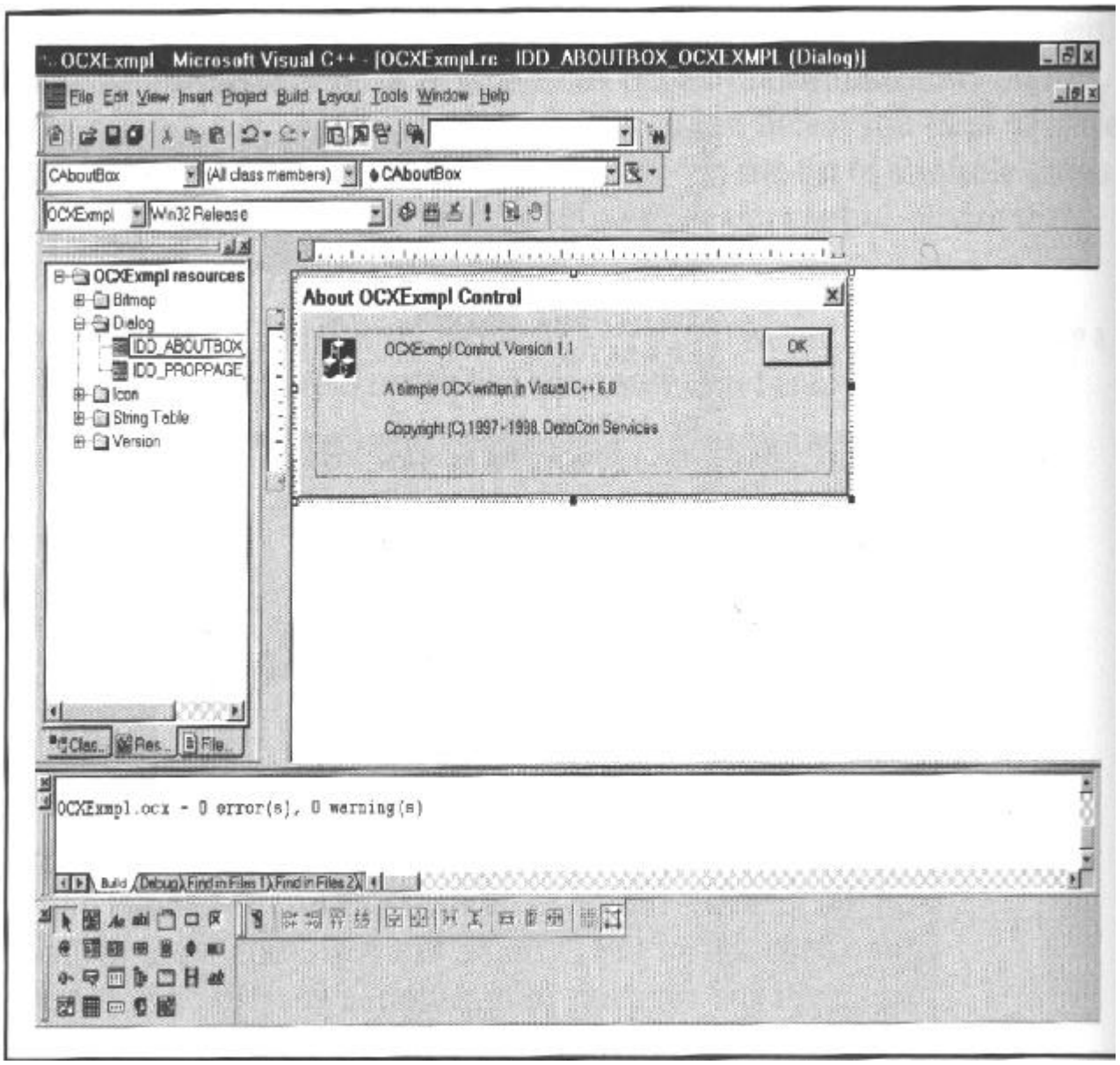
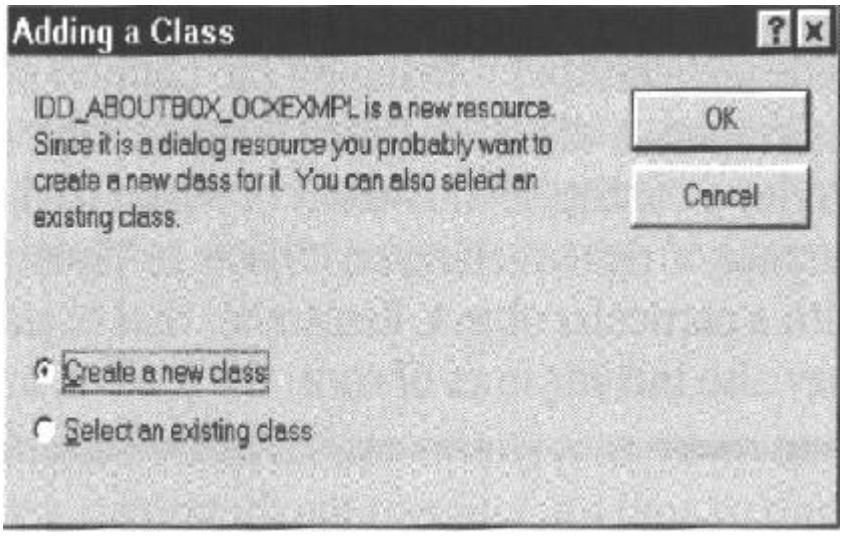
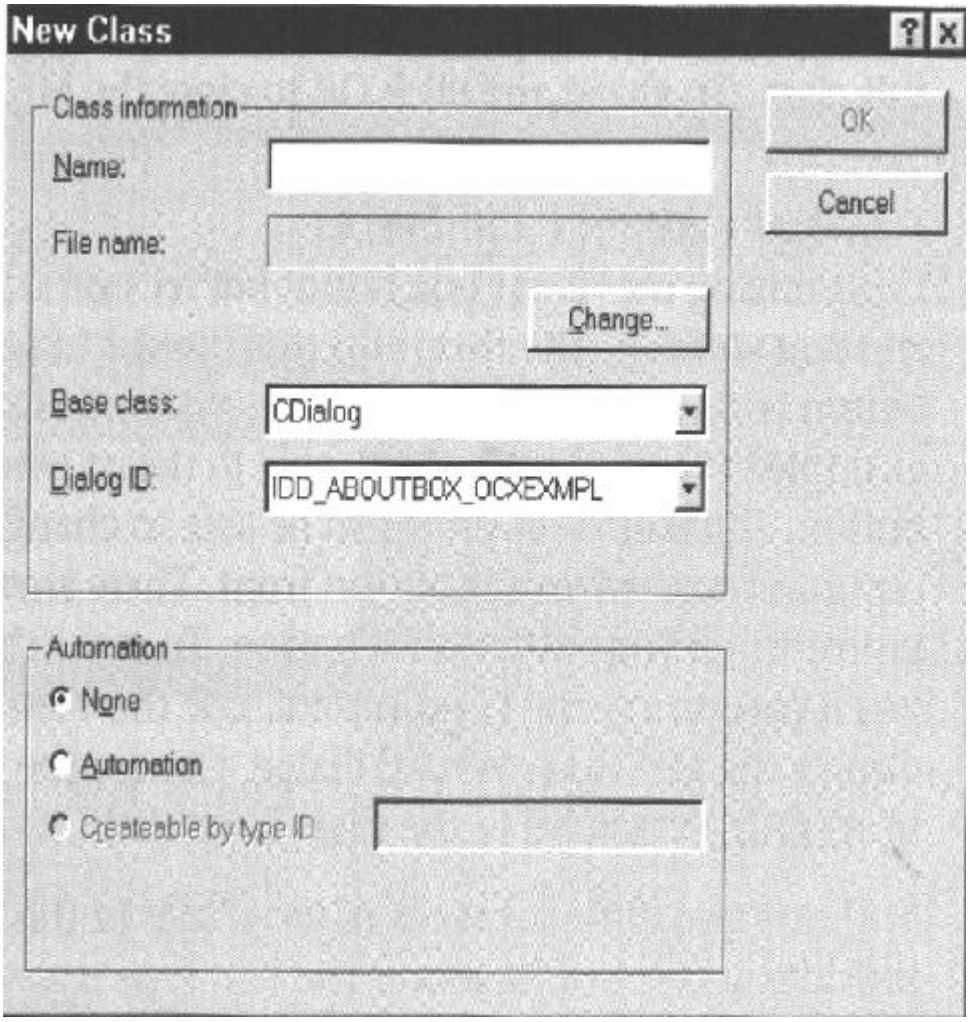


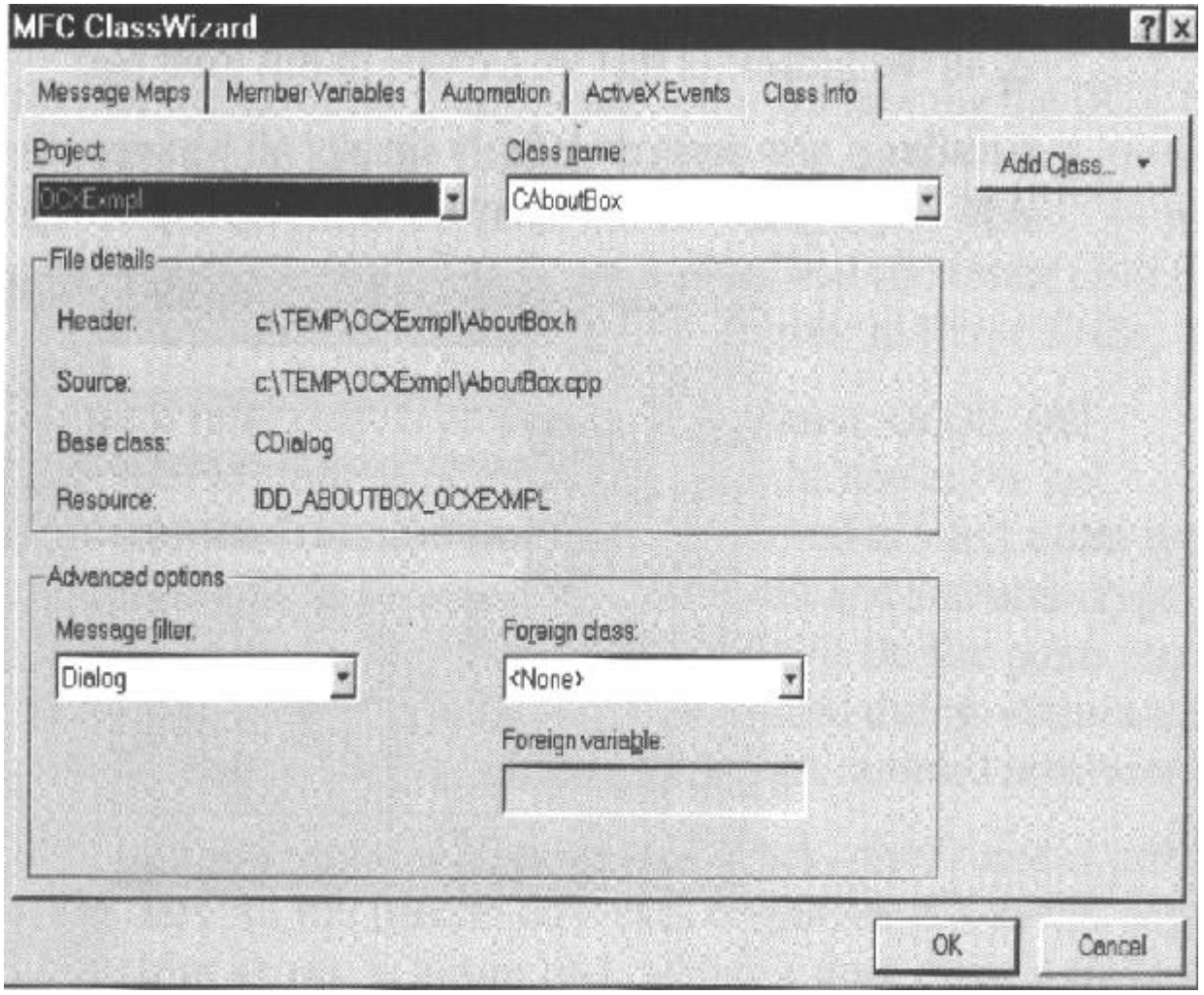
图 10.1　**该对话框编辑器看起来和** Visual Basic **的一样，区别在于你必须在主编辑器屏幕中单独访问它**



选择 Creat a New Class（**创建新类**）**选项，单击** OK。Visual C+ + **则显示如下的** New Class（**新类**）**对话框。**



我们这儿用 CAboutBox **作为类名，把它键入到** Name**（名称）域。这时这个例子所需的一切都准备好了。单击** OK**，则创建所需的类。这时，就会看到在** MFC ClassWizard **（类向导）对话框中出现了一个新项，如下图所示（如果屏幕显示不是这样，那么选一下** ClassInfo **选项卡即可）。**



这个对话框把新类的有关信息都显示出来。Header（**头文件**）**及** Source（**源码**）**两项是与这个类相关的文件的存放位置。你还会看到，这个类的对话框资源是** IDD_ABOUTBOX_OCXEXMPL**。单击** OK**，关闭** MFC ClassWizard(**类向导**）**对话框。**

添加属性及事件

定制对话框的工作很有趣，现在就让我们继续创建 OCX 吧。首先要完成的工作是使按钮控件的某些属性和事件让使用该 OCX 控件的人能够访问（即可见）。例如，在用户单击按钮时要能检测到这个事件。肯定你想改变缺省的属性，如按钮正面上显示的标题等。第一次创建这个按钮时，没有多少属性是可见的。为了使这些元素成为可见状态，需要使用 Class Wizard（类向导）。使用 View（视图）|ClassWizard（类向导）命令，系统显示 MFC ClassWizard（类向导）对话框。选取 Automation（自动化）这一页，然后在 Class Name（类名）域选择 OCXExmplCtrl。

注 Visual C++对你创建的控件作的假设非常少，甚至于使用什么属性和方法使控件可见也不做任何假定。

在这个例子中，使用了两类不同的属性（实际上 Microsoft 提供了许多供你使用的属性）。第一种是常备(Stock)属性。你会发现，第一次创建 OCX 时，就连我们都认为是理所当然的 Caption（标题）属性等等一类事物都是不可见的。常备属性（在图 10.2 中用 S 表示）是由父类缺省支持的一种属性。另一种是定制属性（在图 10.2 中用 C 表示）。定制属性是把一个特定类作为子类时你向它添加的属性，其中之一是我们将用于创建一个 OnOff 控件的 OnOff 属性。本章下面来讨论做这些事的过程。图 10.2 显示了创建我们例子时所有属性的完整列表。

注 常备属性是基类的一部分，定制属性是把控件作成子类时创建的属性。

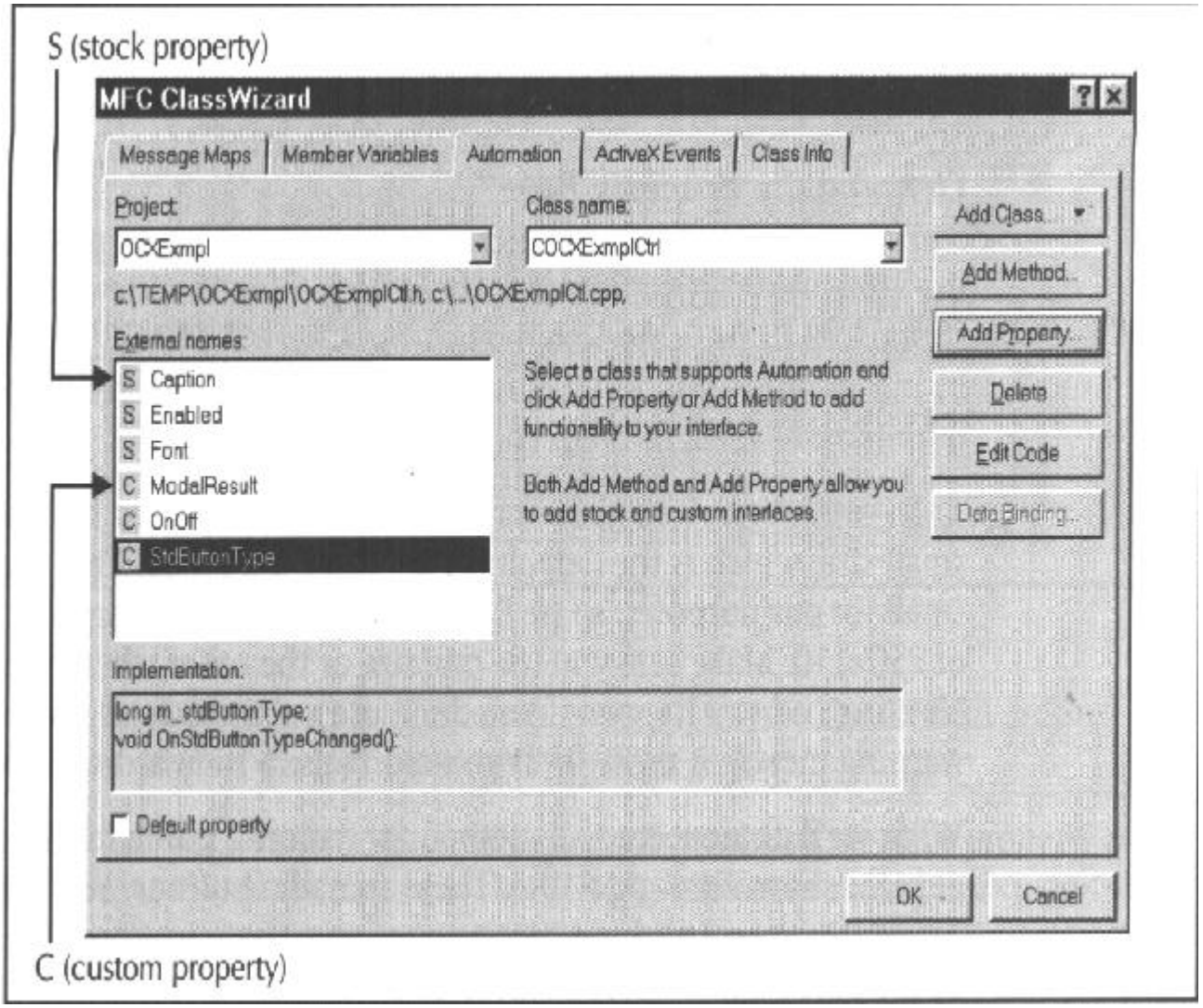
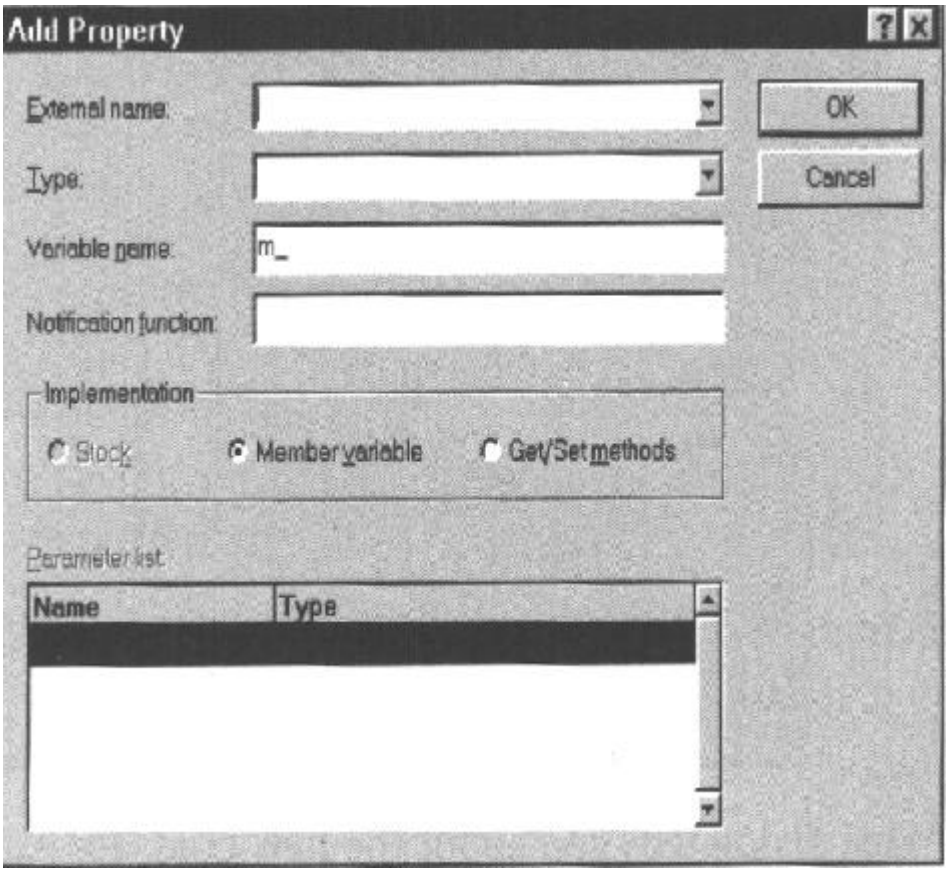


图 10.2　MFC ClassWizard**（类向导）使属性及事件对** OCX **用户成为可见状态**

创建新属性的过程很简单。只要单击一个 Add Property**（添加属性）按钮就可以显示出下图所示的** Add Property **对话框。**



Add Property（添加属性）对话框中包含一些以前我们未曾见到过的重要特性。External Name 组合框包含了创建 OCX 时选择的基类的全部缺省属性，在我们的例子中，就可见到 Caption 属性等。为了创建常备属性，从这个列表中选一项并单击 OK 即可，Visual C++将处理细节。现在接着创建出本例子程序中的全部常备属性（参见图 10.2）。

我们还需要三个定制属性：ModalResult、OnOff 及 StdButtonType。为了创建它们，把它们键入到刚才提到的 External Name（外部名称）域中。然后需要在 Type （类型）字段中为它们选择数据类型。在我们例子中，ModalResult 和 StdButtonType 属性是 long 型，而 OnOff 是 BOOL 型。（在这里，你可能要先

关闭 MFC ClassWizard（类向导）对话框，然后再打开它，否则在有些时候，你选择的属性可能没记录下来。）

这个例子中我们使用的事件都是常备的，他们来自于按钮基类的一部分。要做的是单击 ActiveX Events（事件）页，系统显示如图 10.3 所示。添加常备事件的做法与添加一个常备属性的方法一样，只需单击 Add Event（添加事件）按钮显示出 Add Event（添加事件）对话框，从 Extenal Name（外部名称）组合框中选取一个常备的名称，然后单击 OK 即可。图 10.3 显示了需要为这个例子选择的全部常备事件。

定义属性页

现在该向属性页中添加一些功能了，访问属性页的方法与访问 About 对话框的方法相同，就是使用 View（视图）|Resource Symbols（资源符号）命令。在 Resource Symbols（资源符号）对话框选择 IDD_PROPPAGE_OCXEXMPL 项。属性页用途广泛，其中绝大多数是面向配置的。

Visual C++支持两种标准规格的属性页，OCX 的缺省选择是 250×62 的小号规格。这种规格对我们的目的说来太小了，所以需要把它变成 250×110 的大号属性页。创建一个控件时一定要确定用这一种还是另一种。你不用标准尺寸时也不会发生什么坏事，但用户可能会收到警告信息，说你没使用标准尺寸的属性页。

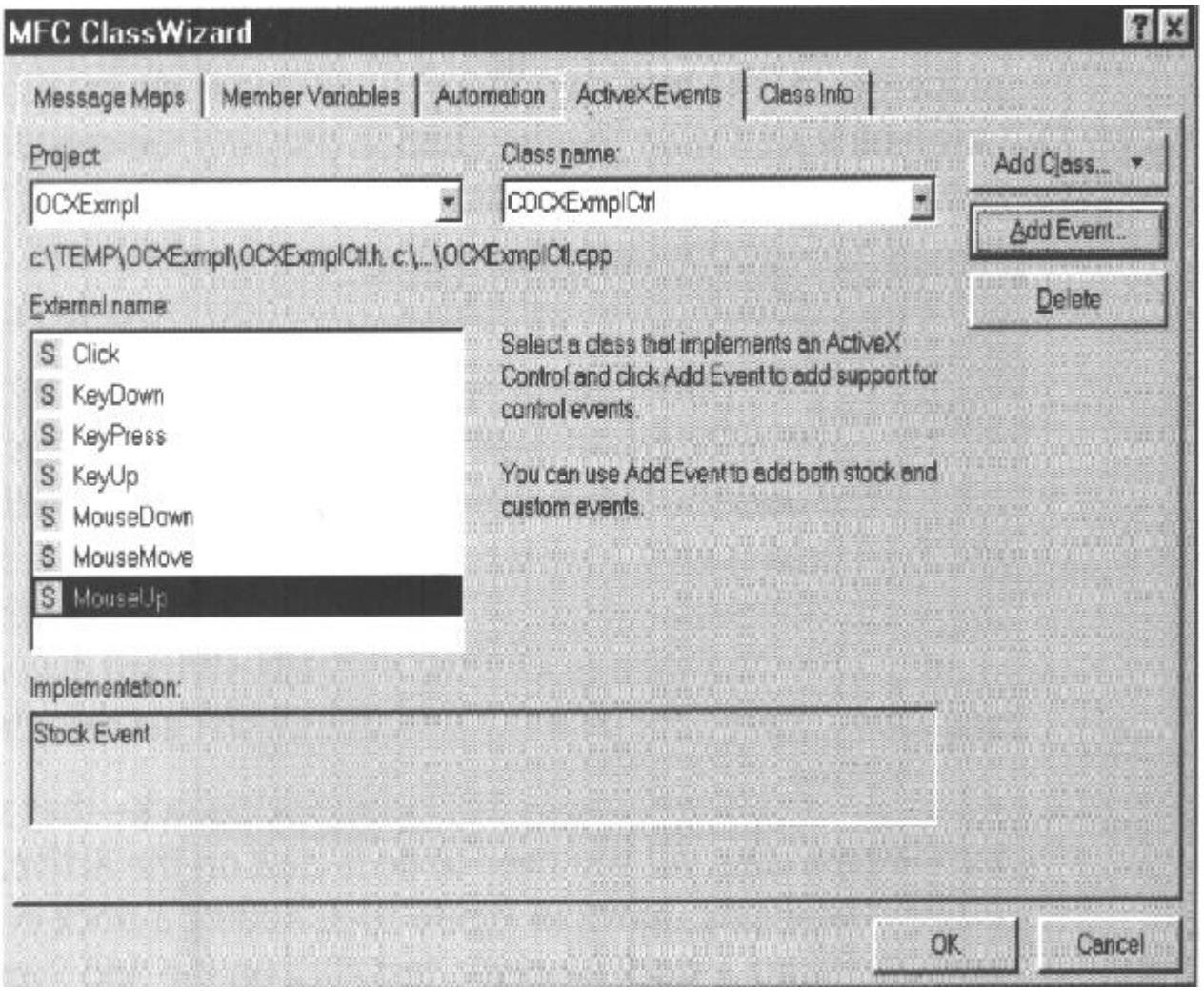


图 10.3　ActiveX **事件页显示了在我们的** OCX **程序例子中要添加的全部常备事件**

现在要完成的工作是，为向页中定义标准的按钮类型来添加一种方法，如图 10.4 **所示。它们是单选按钮，你需要** 10 **个按钮（现在不必担心如何配置它们，下述段落中会进行讲解）。每个单选按钮应有不同的** ID**，以便于能检测出用户单击的是哪一个。（参见** Radio Button Properties**（单选按钮属性）对话框**

中 General 页上的 ID 域）。

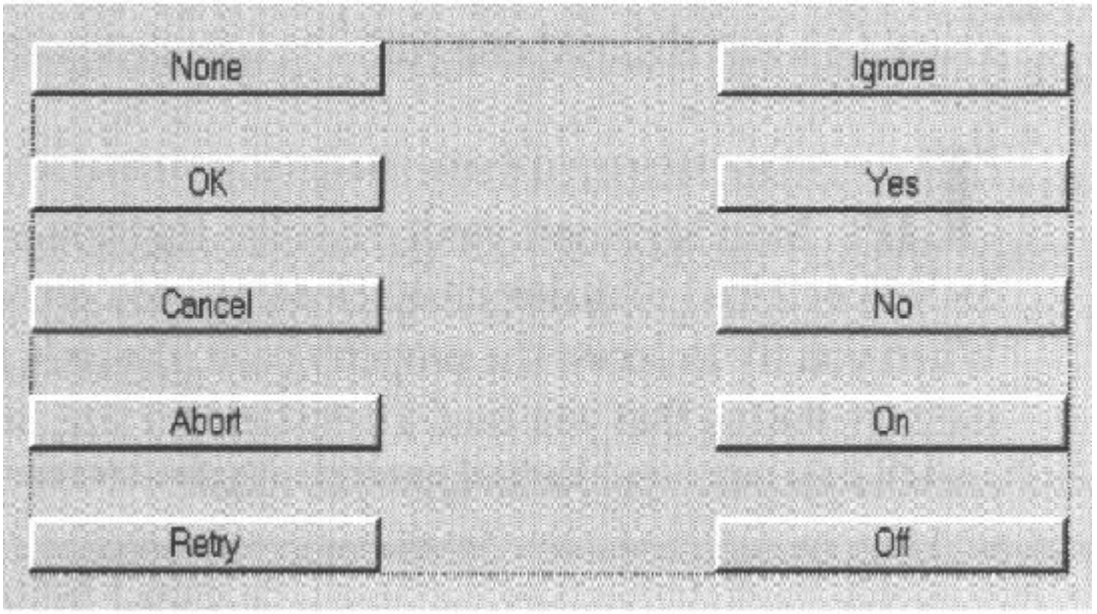
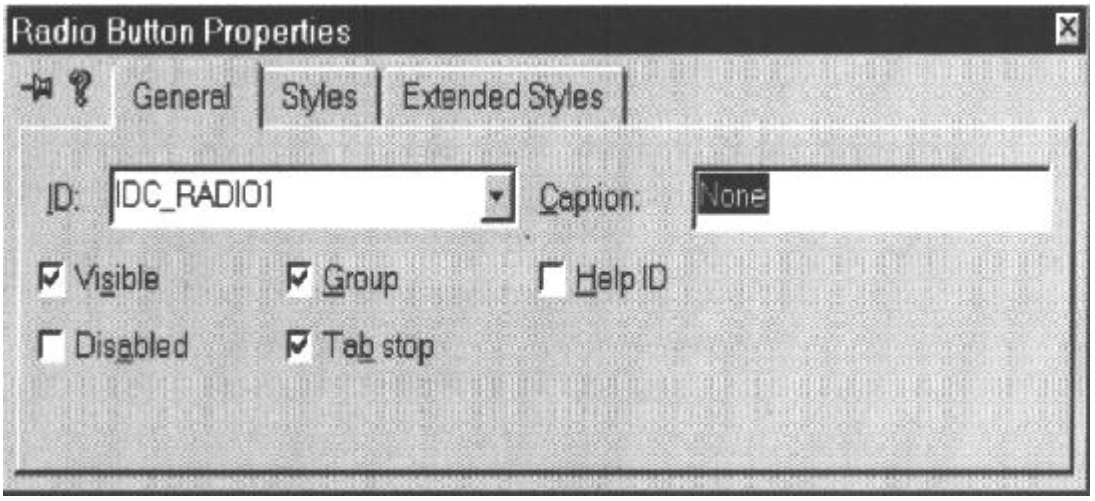


图 10.4　Property Page（属性页）对话框允许用户创建除 on/off 按钮之外的标准按钮类型

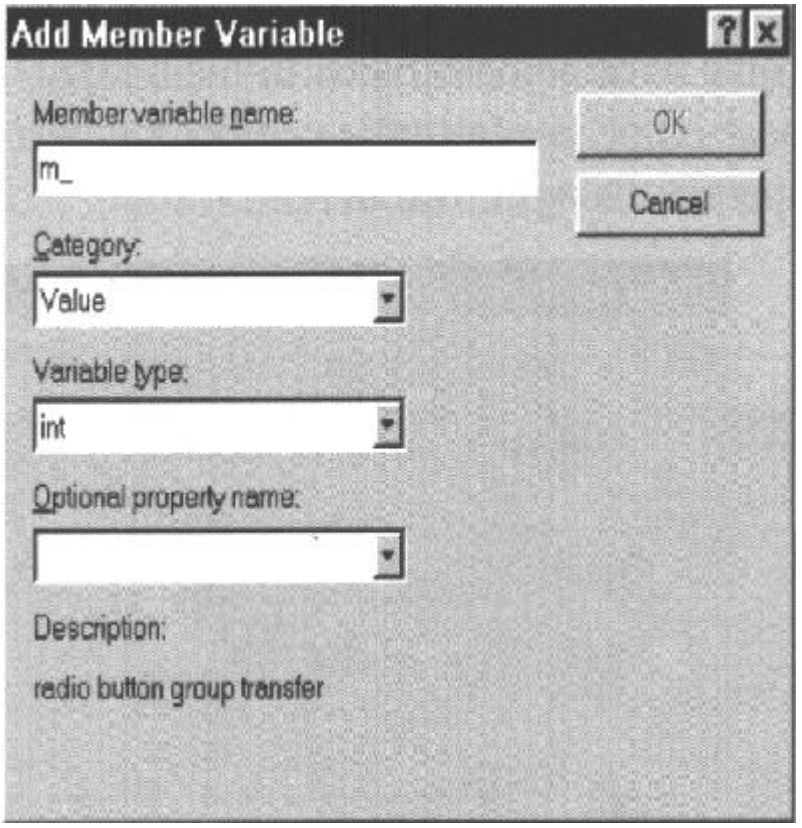
属性页上已有了 10 个标准的单选按钮，现在让它们完成一些任务。用右键单击其中一个单选按钮，从上下文菜单中选择 Properties（属性），系统显示 Radio Button Properties（单选按钮属性）对话框，如下图所示。



为了使单选按钮看起来与如图 10.4 所示的外观相同，要对它们作些小的改动。首先，在 Radio Button Properties（单选按钮属性）对话框选择 Style（风格）页，并为每个按钮选中 Push-like 复选框。还需把这些单选按钮放入一个组，使得当你选择一个新按钮时，当前选择能消除掉。为了完成这个任务，先对使用 IDC_RADIO1 缺省 ID 的组中的第一个单选按钮，复选 Group 及 Tabstop 复选框，而对（该组中）其它的单选按钮都只复选 Tabstop 复选框。否则，最后得到的是每组只有一个按钮的 10 个组而不是有 10 个按钮的一个组。Visual C++把它看到的第一个选择了 Group 复选框的按钮作为组的起点，顺序下去，在 Visual C++看到选择了 Group 复选框的下一个按钮之前，所有按钮都归为这一个组。

技巧 大多数 Microsoft 产品推荐使用尺寸为 250×62 或 250×110 对话框单元的属性页，但你可以按照需要使用任意尺寸的属性页，当你试图访问这个属性页时，你就会看到一条消息，说你没使用标准尺寸。清除这一消息，则属性按通常那样显示出来。

在这个对话框中，我们要对单选按钮再做一件事。为了在单选按钮和 OCX 控件之间建立 OLE 连接，要对它们的输出赋以 OLE 属性。按下 CTRL 键并双击该组中第一个单选按钮(None)，系统显示 Add Member Variable（添加成员变量）对话框，如下图所示：



技巧 键入 CTRL-W 显示 MFC ClassWizard 对话框，选择 Member Variable（成员变量）页，再单击 Add Variable（添加变量）按钮，也可以访问这个对话框。

这儿定制的各个项目都是十分关键的，原因在于 Visual C++不检查它们的

对错，并且也不能从列表中选择它们。在 Member Variable Name（成员变量名）域，键入 m_stdButtorType，这是前面我们创建的定制属性之一的内部名称。保持 Category（范畴）域及 Variable Type （变量类型）域不变。在 Optional Property Name （可选属性名称）域中，键入 StdButtonType，这是把属性页链接到你的 OCX 控件的一个项。记住 C++是区分大小写的，所以大写很重要。

技巧 Optional Property Name（可选属性名称）域的下拉列表中通常包含了从控件基类继承来的属性的完整列表。

添加代码

直到现在，我们还没有向我们的应用程序中添加一行代码。因为我们一直在忙于建立代码框架。现在该向 OCX 中添加代码了。首先要添加一些代码，使得我们的控件能与使用这个控件的用户交换数据，例如，当看到控件属性对话框时，通常想看到这些属性的当前值，类似地，当改变了一个属性值时，就想确信实际控件也随之改变。程序列表 10.1 是需要添加的代码。

程序列表 10.1

```
VOID COCXExmplCtrl::DoPropExchange(CPropExchange* pPX)
{
    //Default actions on the part of the Class Wizard.
    ExchangeVersion(pPX,MAKELONG(_wVerMinor,_wVerMajor))
    COleControl::DoPropExchange(pPX);
```

```
    //Make all of our properties persistent.
    PX_Bool(pPX,"OnOff",m_onOff,FALSE);
    PX_Loog(pPX,"ModalResult",m_modalResult,mrNone);
    PX_Loog(pPX,"StdButtonType",m_stdButtonType,0);
}
```

现在假定你不喜欢按钮的缺省大小，并且想在用户把按钮插入到 Web **页或其它框架中时，显示一个具体的标题。可以在** OnReset()**函数中改变这两个属性。程序列表** 10.2 **中显示了需要修改的代码。请注意，我们使用** COleControl **类的函数来完成所需的修改。**SetText()**函数修改按钮的标题。每当用户插入这一控件时，标题“** Button **”就会显示在该控件上。**SetControlSize()**函数把控件大小变成** 75×25 **个像素。显然，可以按你自己的意愿设置这些属性，甚至能够选择其中一个按钮作为缺省按钮。**

程序列表 10.2

```
VOID COCXExmplCtrl::OnResetState()
{
    COleControl::OnResetState();    //Resets defaults found in DoPropExchange

    //Modify the Microsoft control to match custom size settings.
    COleControl::SetText("Button");
    COleControl::SetControlSize(75,25);
```

```
}
```

现在我们有了交换信息的方法，并且把控件按我们的要求进行了设置。下面来实现我们创建的三个定制属性。是的，每当你创建一个定制属性时，就需要定义一些代码，使得这种属性能完成一些任务。否则，它就只能呆在那里干不成事。程序列表 10.3 列出了用以实现 ModalResult、OnOff 和 StdButtonType 属性所需添加的代码。下一节我来解释一下这些代码的内部操作方式。在目前，你只要知道这些代码实现了我们创建的属性就够了。

程序列表 10.3

```
void COCXExmplCtrl::OnModalResultChanged()
{
    //We don't need to do anything here except set the modified flag.
    SetModifiedFlag();
}

void COCXExmplctrl::OnOffChanged()
{
    //If the programmer set the OnOff property true,take appropriate action.
    if (m_onOff)
    {
        COleControl::SetText("On");     //Change the Caption
```

```
            m_SetOn = TRUE;                 //Set an internal caption flag
            m_modalResult = mrOn;           //Set the modal result value.
        }
        else
        {
            COleControl::SetText("Button");     //Restore default Caption
            m_SetOn = FALSE;                    //Turn our caption flag off
            m_modalResult = mrNone;             //Use the default modal result.
        }

        //Perform the default action.
        SetModifiedFlag();
}

void COCXExmplCtrl::OnStdButtonTypeChanged()
{
   // Change the modal result and button caption to match the user selection.
        switch (m_stdButtonType)
        {
        case 0:
            m_modalResult = mrNone;
```

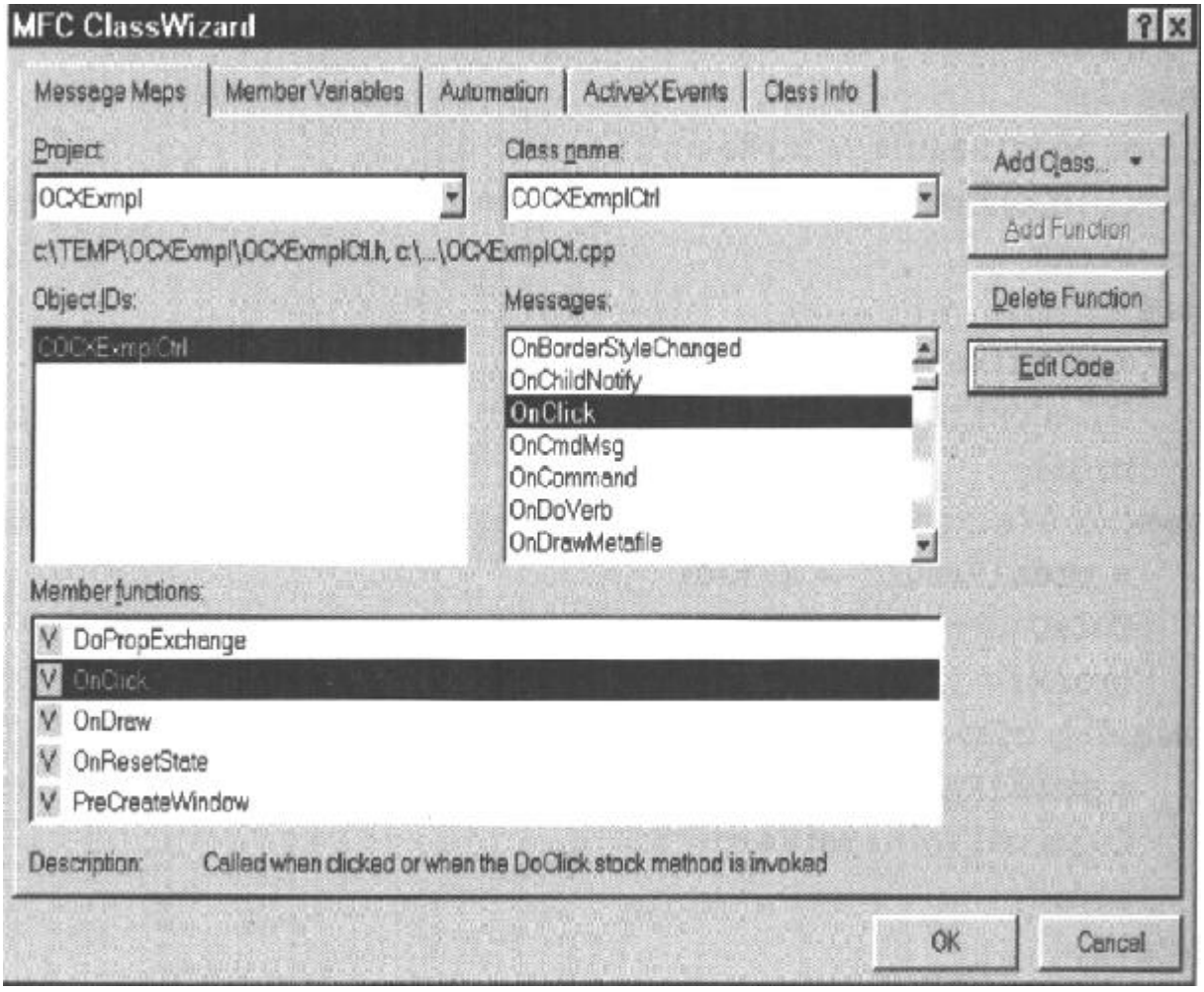
```
        COleControl::SetText("Button");
        break;
case 1:
        m_modalResult = mrOK;
        COleControl::SetText("OK");
        break;
case 2:
        m_modalResult = mrCancel;
        COleControl::SetText("Cancel");
        break;
case 3:
        m_modalResult = mrAbort;
        COleControl::SetText("Abort");
        break;
case 4:
        m_modalResult = mrRetry;
        COleControl::SetText("Retry");
        break;
case 5:
        m_modalResult = mrIgnore;
        COleControl::SetText("Ignore");
```

```
        break;
case 6:
        m_modalResult = mrYes;
        COleControl::SetText("Yes");
        break;
case 7:
        m_modalResult = mrNo;
        COleControl::SetText("No");
        break;
case 8:
        m_modalResult = mrOn;
        COleControl::SetText("On");
        break;
case 9:
        m_modalResult = mrOff;
        COleControl::SetText("Off");
        break;
}

//Set the OnOff property to false since the user selected another type.
m_onOff = FALSE;
```

```
    //Set the modified flag.
    SetModifiedFlag();
}
```

我们现在需要在 OCXEXMPLCtl.cpp **文件中做最后的一些编码工作。当用户单击按钮时怎么办？如果他使用的是标准按钮类型之一，**OnOff **控件将返回一个标准的模式结果值。但是，**OnOff **控件还有一个特殊行为。如果你把** OnOff **属性设置为** True**，按钮则随着用户的单击而在** On **和** Off **之间切换。我们需要添加一些特殊的事件代码来处理这种情况。使用** View**（视图）**|ClassWizard**（类向导）命令显示出** MFC ClassWizard **（**MFC **类向导）对话框，选取** Message Maps**（消息映射）页，选择** Class Name **（类名）域中的** COCXExmplCtrl **项，在** Messages**（消息）列表中选中** OnClick**，单击** Add Function**（添加函数）向类中添加一个函数框架，这时** MFC ClassWizard**（**MFC **类向导）对话框如下图所示：**



现在向 OnClick()**函数中添加代码。单击** Edit Code**（编辑代码）按钮，**Visual C++**就把你带入到新函数中。程序列表** 10.4 **列出了需要添加的代码。**

程序列表 10.4

```
void COCXExmplCtrl::OnClick(USHORT iButton)
{
    //see if the OnOff flag is set. If so, change the caption and internal
```

```
        //caption flag.The effect you should see from this code is a toggling
        //of the caption text.
        if (m_onOff)
        {
                if (m_SetOn)
                {
                        COleControl::SetText("Off");
                        m_SetOn = FALSE;
                        m_modalResult = mrOff;
                }
                else
                {
                        COleControl::SetText("On");
                        m_SetOn = TRUE;
                        m_modalResult = mrOn;
                }
         }
         //Call the default OnClick processing.
        COleControl::OnClick(Button);
}
```

现在我们已经实现了函数编码部分，但我们还需要向 OCXEXMPLCtl.h **文**

件中添加两个支持项。第一个项是枚举类型，其唯一目的是使源代码容易阅读。每个元素对应一个标准按钮类型。第二个项是一个特殊变量。如果注意一下代码，就会看到我们一直引用一个名为 m_SetOn 的成员变量，但这个变量现在还不是类的一部分。程序列表 10.5 显示了如何向头文件中添加所需的枚举类型及特殊变量——就在 Event maps 和 Dispatch and event IDs 项之间添加。

程序列表 10.5

```
//Event maps
    //{{AFX_EVENT(COCXExmplCtrl)
    //}}AFX_EVENT
    DECLARE_EVENT_MAP()
    //Creat a new enumerated type for the modal result.
    typedef enum
    {
        mrNone = -1L,
         mrOk = 1L,
        mrCancel = 2L,
        mrAbort = 3L,
        mrRetry = 4L,
        mrIgnore = 5L,
        mrYes = 6L,
        mrNo = 7L,
```

```
        mrOn = 8L,
        mrOff = 9L,
}MODALTYPE;

//Special On/Off state variable.
    BOOL    m_SetOn;

//Dispatch and event IDs
public:
        enum{
        //{{AFX_DISP_ID(COCXExmplCtrl)
        dispidModalResult = 1L,
        dispidOnOff = 2L,
        dispidStdButtonType = 3L,
        //}}AFX_DISP_ID
        };
```

详细解析代码

你对上述代码作出的最初反应肯定是觉得太可怕了，但是，如果每次处理一个函数，这实际上又相当容易。事实上，作为 ActiveX **控件定义处理过程的一部分，**Visual C++**已为你写出了大部分的代码，所以你写出的代码并没有比**

为一个标准应用程序所写的代码多多少。我们添加的函数，只不过是定义我们要求这个控件所要完成的特殊任务。

让我们把代码分解开来看一下。你修改的第一个函数是 DoPropExchange()。这个函数在这个例子中只完成一种服务——把你的定制属性变成持久属性。实际上，PX_ 系列函数调用允许把一个特定属性的值，从一个会话期存贮到下一个会话期。对你定义的每一个变量类型都调用一次该类型的函数。每次调用都接受类似于下文所述的四个变量：

```
PX_Bool (pPX,"OnOff", m_OnOff,FALSE);
```

第一个变量是指向属性交换结构的一个指针，Visual C++自动定义这一结构，你只管使用就是了。第二个参数包含了属性的外部名，用户在 Property Inspector 中可以看到它。（对于 Property Inspector，Visual C++称之为属性对话框，Delphi 称为对象观察器，等等）。第三个参数是属性的内部名。使用它在整个程序中确定该属性。最后的参数是定义属性的缺省值（除非你想让用户在 Properties（属性）对话框中看到一个空白域）。

你修改的下一个函数是 OnResetState()。这个函数提供的是，当用户把组件添加到窗体上时，用户看到的一些美学细节。在这儿，我们给予组件一个缺省标题，并改变其尺寸使之与能在 Web 页上正常工作的定制尺寸相匹配。如果你设计一个 ActiveX 控件用于其它目的，就需要改变这一设置，以满足你使用最多的程序设计语言的需要。要记住重要的是 OnResetState()函数允许你对使用控件所需要的任何设置进行操作。

技巧 Microsoft 使用的组件缺省尺寸差不多是 Borland 产品（如 DelPhi）的两倍。Internet 控件的尺寸可变，但本例中使用的 75×25（宽×高）规格在绝大多数情况下都能工作。

在代码的消息处理程序（handler）部分，三个修改过的函数中有两个需要进一步修改。ModalResultChanged()函数不需作修改，所以这里就不讨论它了。与 ModalResultChange()函数相关的属性 ModalResult 由其它两个函数进行修改。先看一看 OnOffChange()函数。我们要做的是设置内部标题标志和初始标题。如果程序员设置 OnOff 属性为 True，我们通过设置它的标题为 On 把控件设置为一个 on/off 开关按钮。当按钮用作 on/off 开关时，还提供了不同的模式结果值。请留意，内部属性变量 m_onOff 追踪着标志的状态，而内部属性 m_SetOn 追踪着 OnOff 控件的当前状态(on 或 off)，因为这个按钮初始时为 On，所以一开始设置 m_SetOn 标志为 True。

现在看一下这个 OCX 的 Property Page（属性页）中功能所需的处理过程。OnStdButtonTypeChanged 函数只不过是一个 case 语句，它按照创建各种缺省按钮类型的需要来改变按钮的 Caption 和 ModalResult 属性。请留意，如果用户选择了缺省的按钮类型，我们还不得不关闭 OnOff 的按钮处理过程。

OnClick()消息处理函数在运行期间是激活的，这里有两个层次的动作。第一，需要确定程序员是不是把这个按钮定义为 on/off 开关，如果是，我们就改变内部状态变量(m_SetOn)和按钮标题。该函数让按钮状态在 On 和 Off 之间按需要切换。一旦我们完成了使按钮工作所需的内部处理，我们就调用缺省的 OnClick 处理例程。如调用失败，则导致 OCX 跳过你为按钮事件附加的专用于

程序设计环境的代码。例如，如果你在 Visual C++应用程序中使用这个控件，附加于 Visual C++中 exposed 事件的任何代码都会被忽略。

在能够使用这个组件之前，还得在 Visual C++中建立它。建立过程的一部分是用 Windows 自动为你注册 OCX。我确实喜欢这一特色，因为在测试 OCX 时它为我节省了不少时间。唯一的缺憾是，预先注册污染了工作环境，从 Internet 上使用 ActiveX 控件的角度来看，你必须到另一台机器上去测试这一组件。

测试控件

一旦创建了一个任意种类的 ActiveX 控件（不论是否你的编译器是否将它称为 OCX），都要进行一些测试，从而保证它按预期的方式工作。对于想用于 Internet 的 ActiveX 控件来说，其测试工作最好分为四个阶段：三个层次的内部测试和第四层次的外部测试。下述列表说明了各个阶段的重要性。

- 内部测试阶段 1　在标准环境中使用这个控件。在将控件从标准程序设计平台移走之前，看一下它是否能够工作正常。这样做的理由很简单：在浏览器内测试 ActiveX 控件时没有调试程序可用。在把它从 C++（或其它 OCX/ActiveX）程序设计环境中移走之前测试 ActiveX 控件的基本功能，使你拥有调试程序，从而能随时找出真正严重的问题。
- 内部测试阶段 2　在浏览器内进行本地测试。在你的本地机器上建立简短测试，看一看 ActiveX 控件能否装入到一个 HTML 页中，这样可以使后面的工作节省一些时间，你还应该验证一下，是否有足够多的属性在实际使用控件时可用，并且在你浏览测试页时它能否工作。

- 内部测试阶段 3　使用网络连接测试整个 Web 页。一旦你对 ActiveX 控件的基本功能完成了测试工作，并验证了它能在浏览器中正常使用，就要确定它是否能够在一整页 HTML 标记中正常工作。总之，不能与同在一页上的其它控件一起工作的控件有什么用呢？ 控件间的相互作用能引起一些真正奇怪的问题。可以在应用程序中使用标准窗体来测试相互作用问题，但也帮不了多少忙，问题是浏览器并不是像你所喜爱的编译器那样来处理 ActiveX 控件。
- 外部测试　在未被污染的机器上核查控件功能。你遇到的最大问题是污染。请记住，大部编译器总是自动地对你所创建的 OCX 或 ActiveX 控件进行注册。而进入你的 Internet 站点的人却没有这个便利。非常重要的是，用未被污染的客户机和服务器在 Web 页的上下文环境中测试你的新控件。换言之，测试的最后阶段，要像任何访问你的 Internet 站点的人那样来看待这个控件。

现在你已经明白我们要做些什么了，让我们看一下我们创建的控件的测试工作吧。在下面我将使用 Visual C++ 6.0。你可以使用支持 OCX 的任意程序设计环境，例如，可以使用 Visual Basic 或 Delphi 来测试 OnOff 控件，看看在这些语言中控件是如何工作的。重要的考虑不在于你用什么语言来测试，而在于要使用包含了全部调试支持的一些程序设计语言来完全地测试这个控件。在预定要使用控件的最安全的环境中，要确保创建的控件能够真正地按预期方式工作。由于许多 Internet 工具处于测试阶段，你会发现看起来好象控件能工作却又工作不了。在一个已知的环境中，通过对控件进行测试来消除控件本身的干扰，

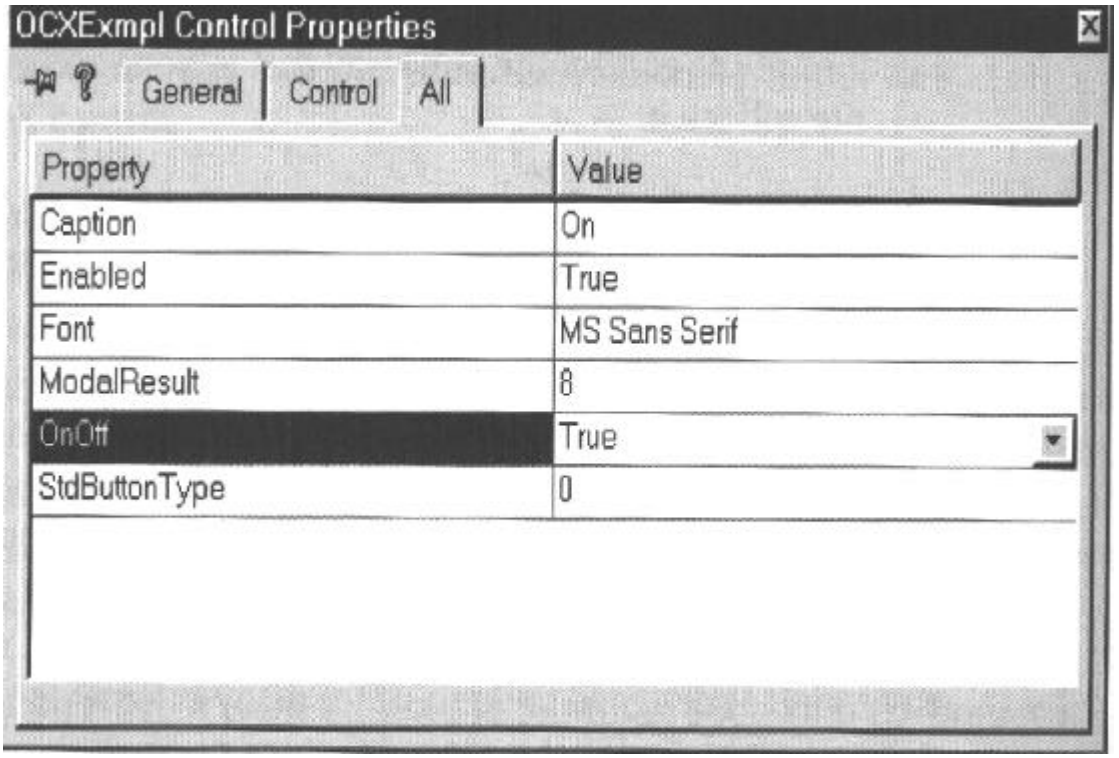
这是迈向找出问题的第一步。

内部测试阶段 1：在标准环境中使用控件

内部测试的第一阶段不会是一个既长又累人的过程。真正需要完成的任务是使用标准程序设计环境创建一个工程，然后把你创建的 ActiveX 控件添加进去。要保证所有属性如预期的那样工作起来，花些时间彻底地核查属性页。

> 注　记住需要使用 Project（工程）|Add to Project|Components and Controls（组件与控件）命令显示出 Components and Controls Gallery（组件与控件展台）对话框，在这个对话框中把 OCXExmpl 控件添加到当前工程中去。

在你喜欢的（支持 OCX 工作的）程序设计环境中创建一个新工程。对于本例来说，在 Visual C++中创建一个基于对话框的应用程序就能容易地测试该控件。MFC AppWizard（MFC 应用程序向导）为你做了大部分工作。（在第 2 章中已讨论过用 Visual C++ 创建基于对话框的应用程序的过程）。我们把这个工程命名为 OCXTest。



创建了新工程之后，必要时再创建一个窗体，并把 ActiveX **控件添加进去。**Visual C++**自动地将** ActiveX **控件注册，所以前一节中创建的控件就会出现在可用控件列表中。（其它程序设计环境可能需要你单独地注册你的** ActiveX **控件）。图** 10.5 **显示了为调试这一例子而创建的测试程序对话框，它还显示了** Property **（属性）对话框中选中的** Control **（控件）属性页。**

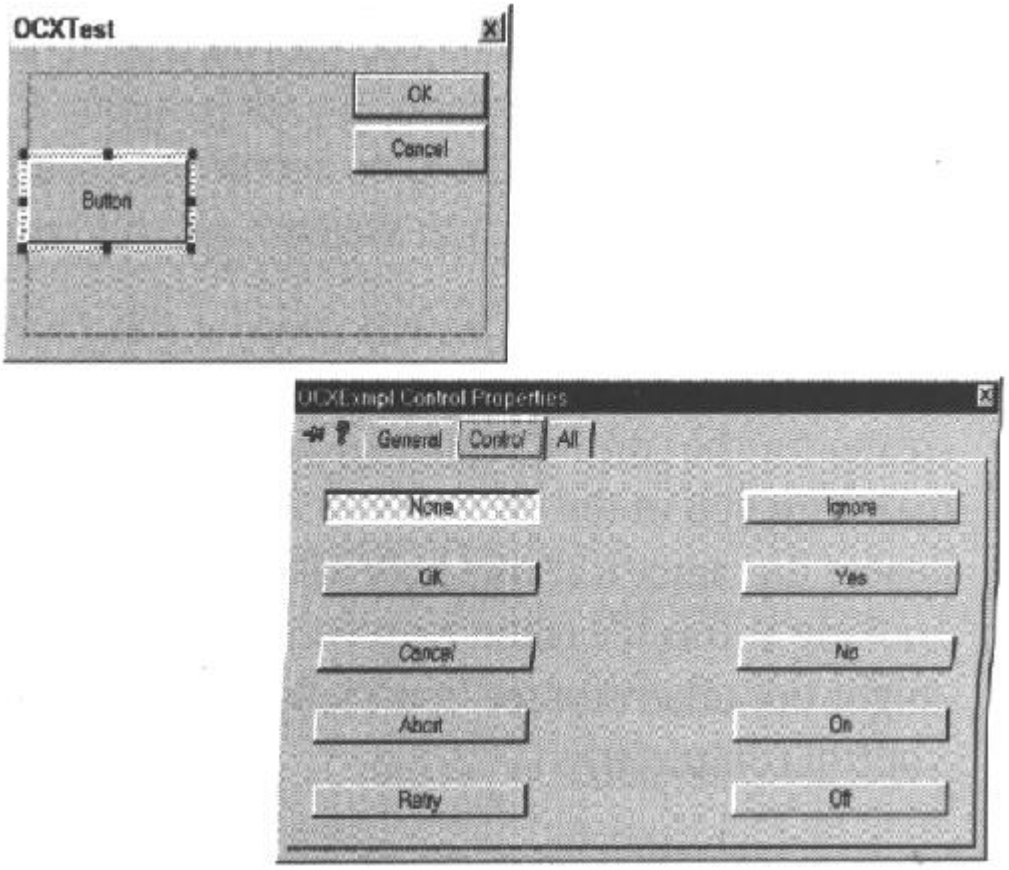


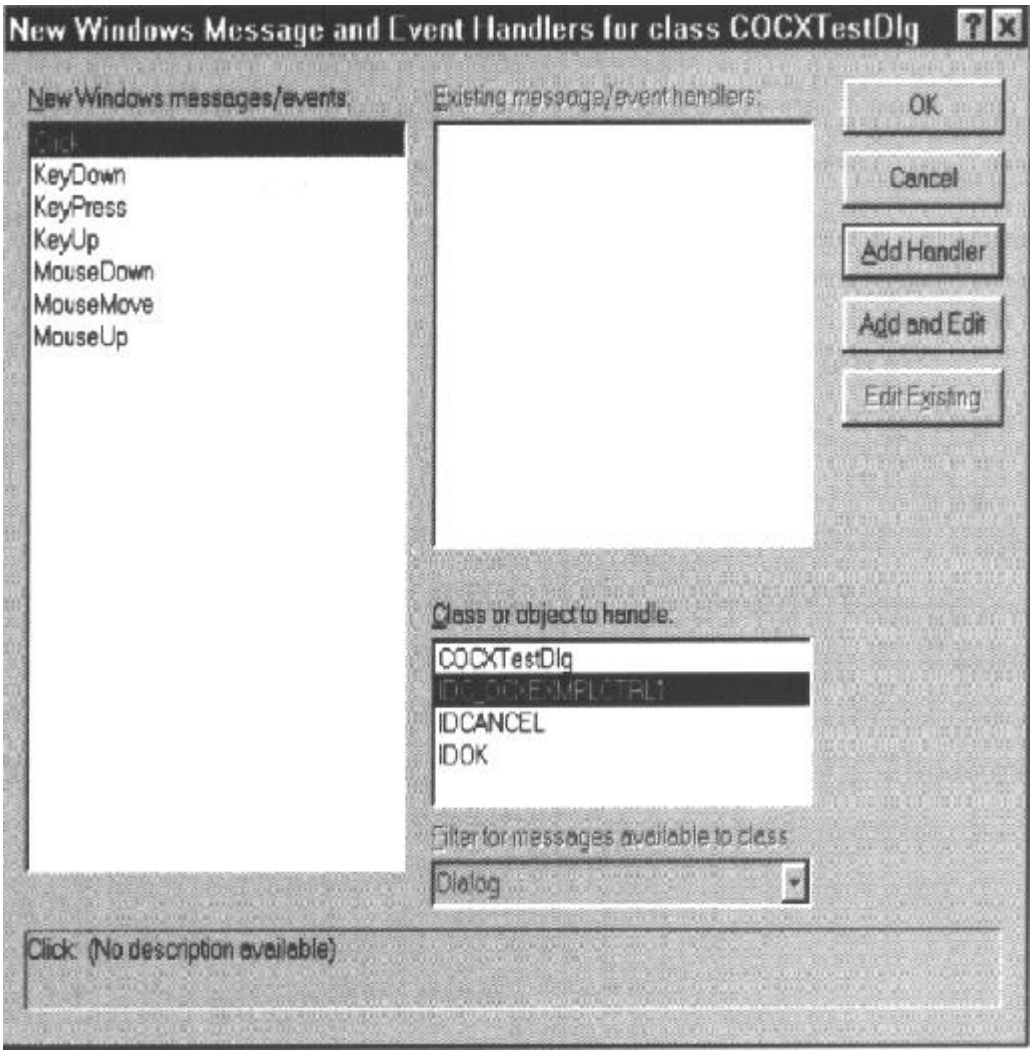
图 10.5　控件属性页对话框表明了这个按钮的特殊特色

绝大多数程序设计环境还提供了一种方法，使得你可以查看与一个控件相关联的所有属性。下图显示了 OCXExmpl Control Properties **（控件属性）对话框的** All **页，其中** OnOff **属性设置为** True**。**

也许你想向程序中添加一些测试代码，从而能够核查各个控件事件的作用。例如，本例中的 On/Off **开关按钮提供了许多依赖于怎样设置按钮属性的模式结果返回值。把** OnOff **属性设置为** True **则创建了一个开关按钮，**ModalResult **属**

性在两个值间切换。但是，也可以从 Properties（属性）对话框的 Control （控件）页中很容易地选择一种标准按钮值。

第一件要做的事是按下 CTRL 键并双击控件，然后看到 Add Member Variable（添加成员变量）对话框，在 Member Variable Name（成员变量名）域中键入 m_OnOffButton，还要保证 Category（范畴）域设置成 Control（控件），并且 Variable Type（变量类型）域设置成 COCXExmpl。现在就能够在测试应用程序中访问控件的属性了。用右键单击控件，在上下文相关菜单中选中 Events，就能看到如下图所示的 New Window Message and Event Handlers（新建窗口消息和事件处理程序）对话框。



请注意，Visual C++ **自动地选择了** Click **（单击）事件和** IDC_OCXEXMPLCTRL1 **对象。需要你做的工作是单击** Add and Edit **（添加与编辑）按钮来向你的程序中添加一个新函数。看到** Add Member Function **（添加成员函数）对话框后，接受缺省函数名并单击** OK**。这时就会看到这个按钮的函数框架。**

程序列表 10.6 **是这个例子的** C++**测试代码。请注意，**Visual C++**自动为控**

件创建的 GetModalResult()封装类函数的使用。在 OCXEXMPLE.H 文件中，可以找到 Visual C++ 为你制作的全部说明。因为头文件说明了 Visual C++是如何与控件打交道的，所以看一下这个头文件是有启发意义的。看一下这个文件，能帮助你找出可能漏掉的接口问题，（特别是如果你不对控件的每个属性进行完全测试时更是如此）。

注释 创建 OnClickOcxexmplctrl1()函数的方法有多种，最简单的方法是使用 MFC ClassWizard（类向导）（使用 View（视图）|ClassWizard（类向导）命令显示这个向导）。在 Object IDs（对象标识）列表框中选中 IDC_OCXEXMPLCTRL1 项，并在 Messages （消息）列表框中选中 Click 项。这时单击 Add Function（添加函数）按钮，就把函数添加到了你的程序中。

程序列表 10.6

```
void COCXTestDlg::OnClickOcxexmplctrl1()
{
    //Get the current ModalResult value
    long liModalResult;
    liModalResult = m_OnOffButton.GetModalResult();

    //Determine which modal result was returned and display a message.
    switch(liModalResult)
```

```
{
case -1:
        Messagebox("None button pressed","State of control",MB_OK);
        break;
case 1:
        Messagebox("OK button pressed","State of control",MB_OK);
        break;
case 2:
        Messagebox("Cancel button pressed","State of control",MB_OK);
        break;
case 3:
        Messagebox("Abort button pressed","State of control",MB_OK);
        break;
case 4:
        Messagebox("Retry button pressed","State of control",MB_OK);
        break;
case 5:
        Messagebox("Ignore button pressed","State of control",MB_OK);
        break;
case 6:
        Messagebox("Yes button pressed","State of control",MB_OK);
```

```
            break;
        case 7:
            Messagebox("No button pressed","State of control",MB_OK);
            break;
        case 8:
            Messagebox("Button is On","State of control",MB_OK);
            break;
        case 9:
            Messagebox("Button is Off","State of control",MB_OK);
            break;
        }
}
```

我们已创建了一个简单窗体并把你的控件添加到该窗体上面，试着测试一下它吧。示例程序将显示一个简单的带有 ActiveX **控件的对话框。单击控件，就会看到另一个对话框，其中显示了按钮的状态，如图** 10.6 **所示。单击** OK **或** Cancel **控钮（由** Visual C++**自动提供）就结束本程序。如前所述，这是对控件基本功能的简单测试。到现在为止，我们核查了属性页、属性和使用控件的结果。**

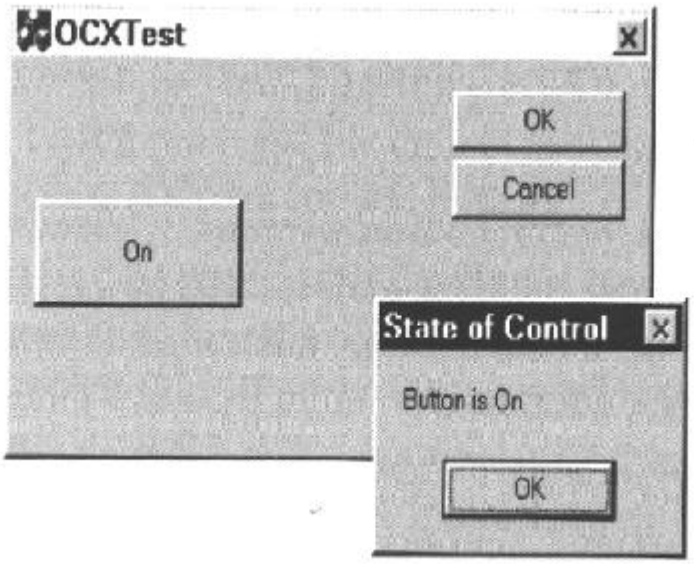


图 10.6　OnOff 控件按预期方式工作：返回模式结果值

内部测试阶段 2： 在浏览器内进行本地测试

到现在为止，我们还没有在本章中做一些哪怕是远远地接近一下在 Web 页中工作的事，至于向 Web 页中添加 ActiveX 控件，涉及得就更少了（在第 8 章中我们确实研究过使用 Web 页问题，如果想了解使用 Internet 的简单情况时，一定去读一下这一章）。该变化一下了，测试过程的下一步就是把控件插入到一个 HTML 文档中，我们再次使用一个非常简单的设置。

注释 在这一节中，我们使用一个非常简单而且容易使用的实用程序。如果计划设计许多普通的或高难度的 Web 页，那么，应该考虑使用某个工具，比如 Visual InterDev 。但 ActiveX Control Pad 用作测试工具或用作设计简单 Web 页更好使用些。

Microsoft 提供了一个 ActiveX Control Pad 实用程序，可用于测试控件时快速建立 HTML 页。图 10.7 是第一次打开这个实用程序时的样子。还可以用它创建功能强大的 Web 页（第 8 章我们讨论过创建过程）。

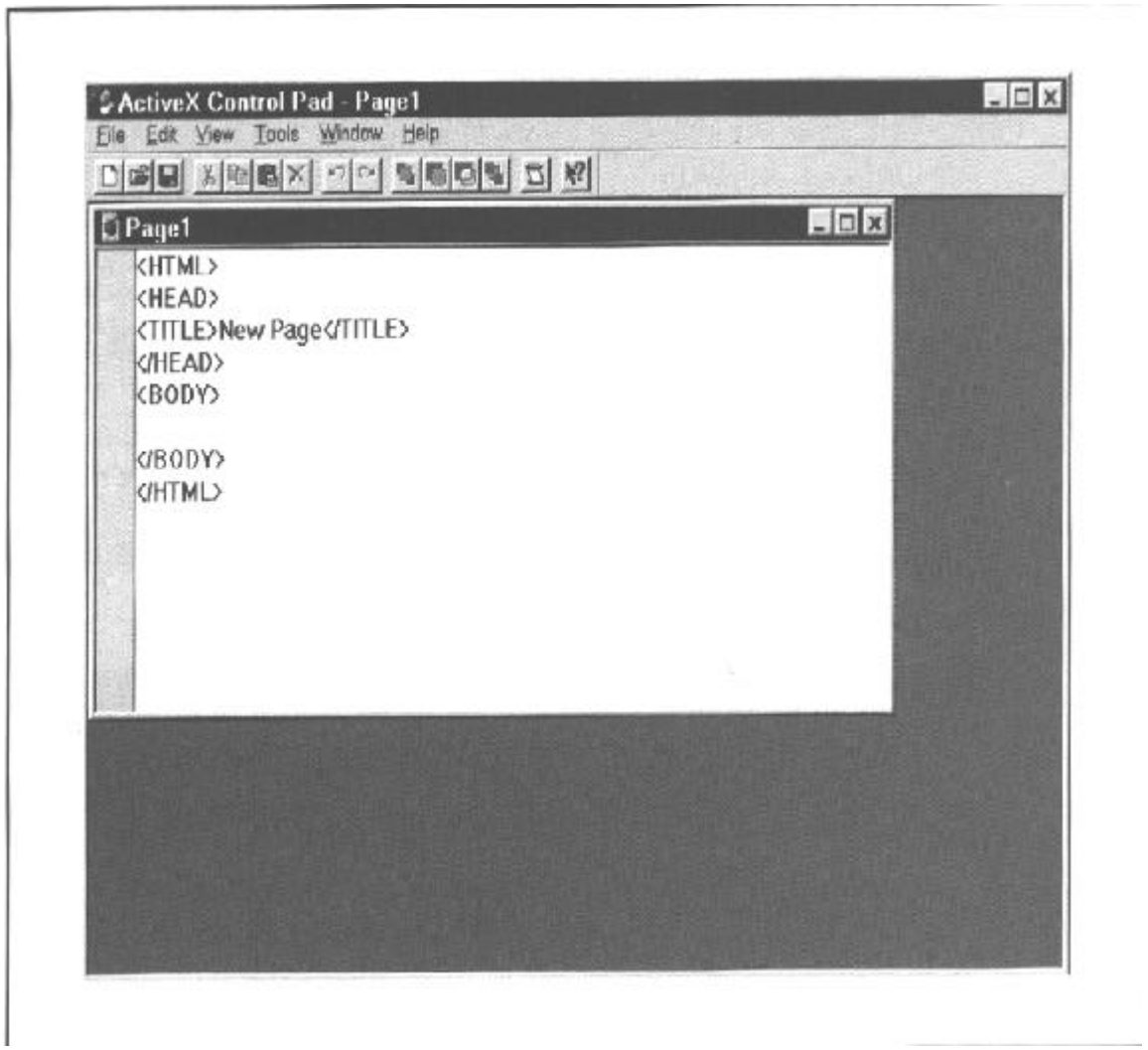


图 10.7　ActiveX Control Pad **允许快速地设计一个** HTML **页来测试你的控件**

为了创建用于测试 ActiveX **控件的** HTML **页，使用** Edit**（编辑）**|Insert ActiveX Control**（插入** ActiveX **控件）命令显示出** Insert ActiveX Control**（插入** ActiveX **控件）对话框。从提供的列表中选择你的控件，单击** OK**，就会看到** ActiveX **控件已装入，如图** 10.8 **所示。正常情况下，应该把该控件放入一个窗体中，但这**

儿唯一的目的只是测试它而已。

每装入一个新控件，ActiveX Control Pad 都会自动地显示出 Properties （属性）对话框，如图 10.8 所示。这个对话框包含了这个控件的已发布的(published)属性的标准列表。请留意图 10.8 还显示出了 General 属性页，通过用右键单击控件并在关联菜单中选择所需的项，就可显示出这个页。对我们的示例来说，要选择第二个 Properties （属性）项。

我们需要改变的唯一一个属性是 Properties(属性）对话框中的 OnOff 属性，把它变成 True，然后关闭包含控件的对话框。ActiveX Control Pad 就向 HTML 页脚本添加一个标记（如果看不懂 HTML 页上的专用术语的话，请到第 8 章找出对 HTML 标记的说明 ）。请注意该项包含了控件的 CLASSID 和所有需要设置的属性。

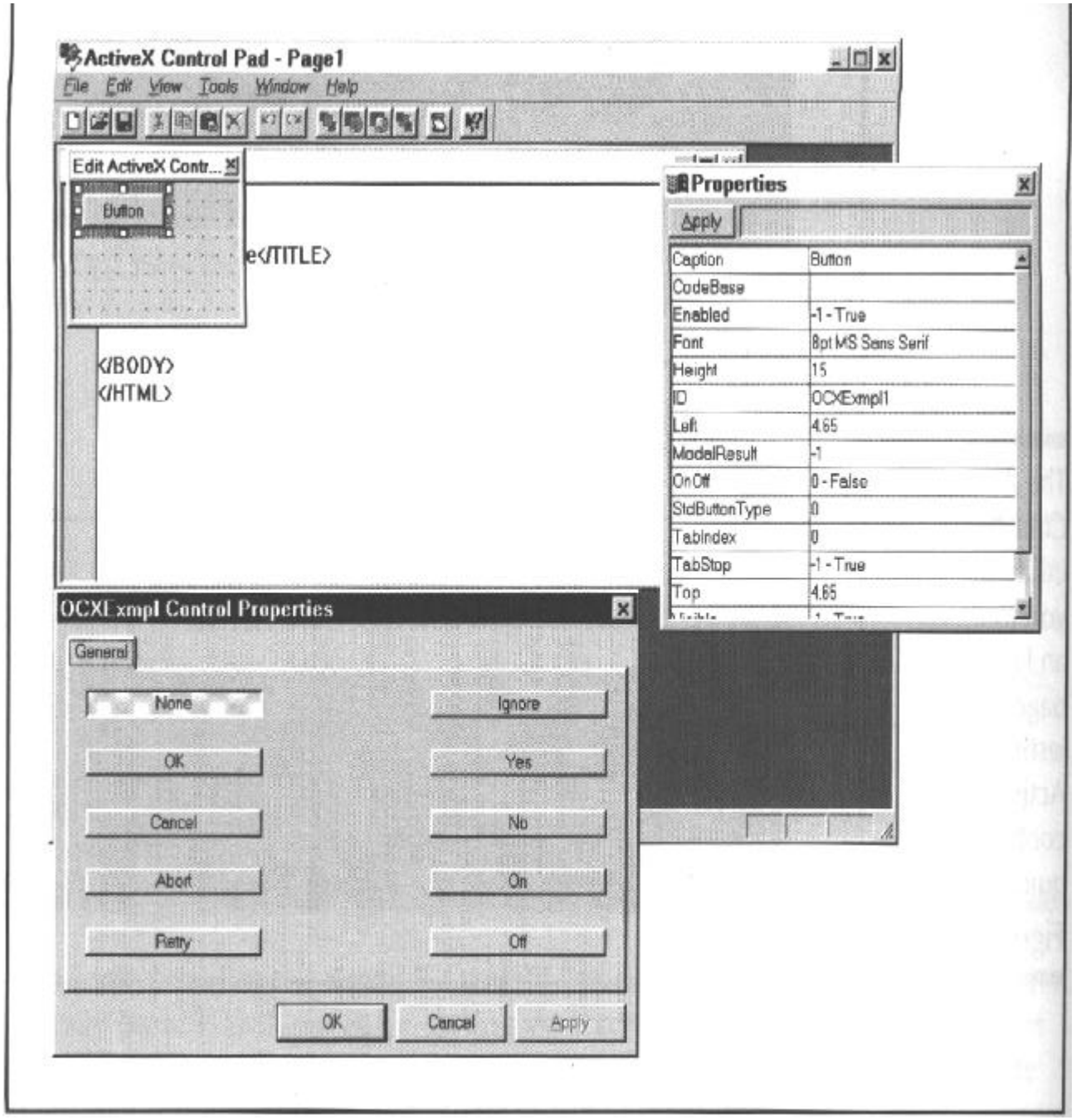


图 10.8　ActiveX Control Pad **在装入你的控件后就显示它**

技巧 你在 HTML 脚本中紧挨着<OBJECT>标记的地方会看到一个小按钮。单击这个按钮，再次显示这个控件，从而可以编辑它的属性。每当你把一个 ActiveX 控件或一个 HTML 页框架放入脚本时，ActiveX Control Pad 就显示这个按钮，使你能容易地按需要编辑控件或页面布局。

使用工具栏上的 Save（保留）按钮保存示例 HTML 页面，我将其命名为 TestPage.HTM 。关闭 ActiveX Control Pad，然后，使用你喜欢的与 ActiveX 兼容的 Web 浏览器（写作本书时，仅限于 Internet Explorer 和使用使用 NCompass ScriptActive 插件的 Netscape Navigator）。打开这个 HTML 页面。如果你使用 Internet Explorer，要保证把浏览器的安全层次设为中等。设置安全层次的步骤是：使用 View（视图）|Options （选项）命令，选择 Security（安全）页，在这个页中单击 Safety Level（安全层次）按钮即可。

注 要保证保存的测试用 Web 页的文件扩展名为 HTM。

装入 TestPage.HTM 文件后，所看到的如图 10.9 所示。现在看上去并不使人肃然起敬，但控件却能做那些标准 HTML 页按钮做不到的事。单击按钮，就会看到标题在 On 和 Off 间切换。你还能监测按钮的模式结果值并确定它处于什么状态，所有这一切，一行 CGI 脚本代码也不用写就做到了！（本章后面将讨论从 ActiveX 控件中得到信息的方法。）

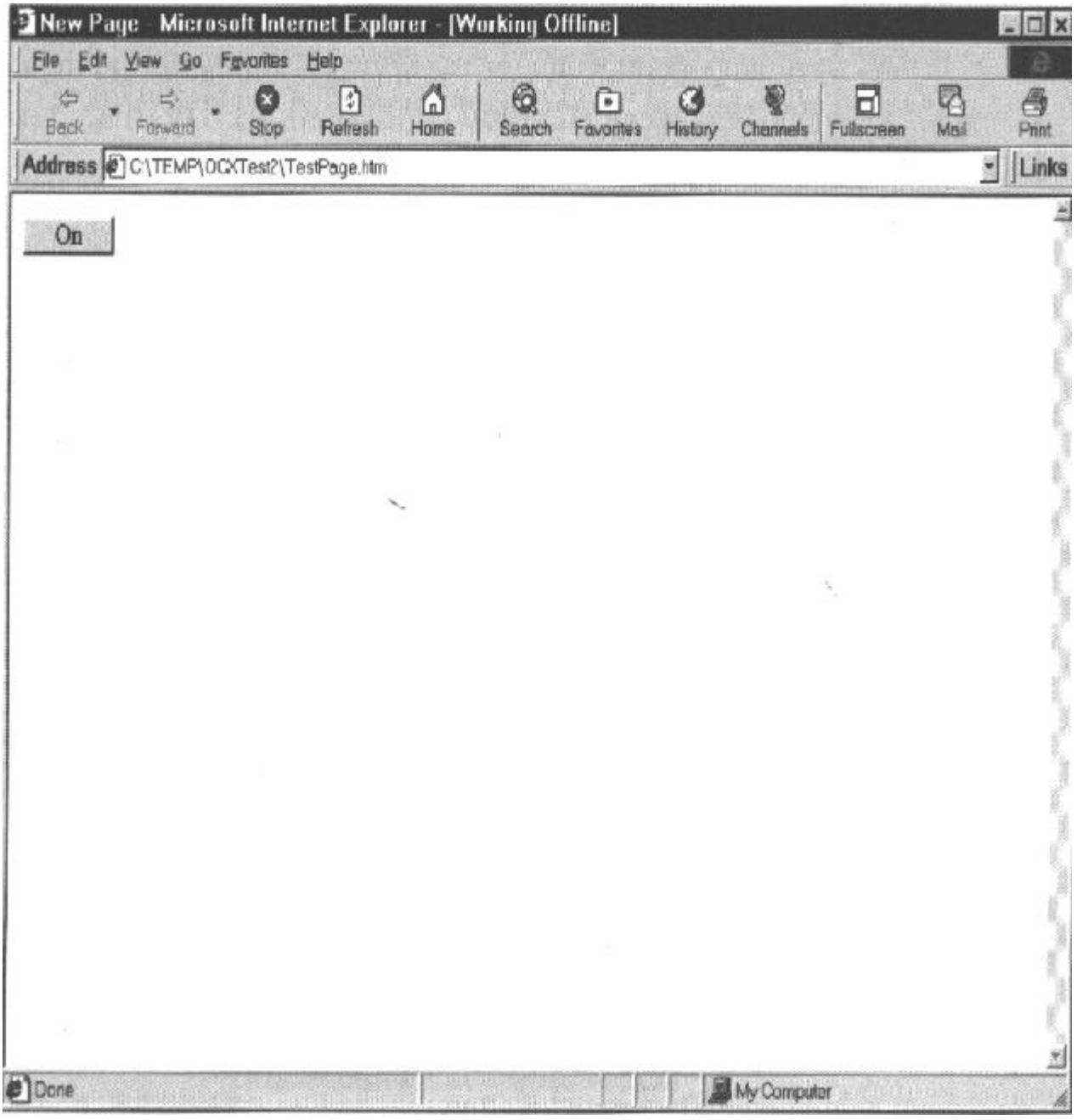


图 10.9　ActiveX **控件在这个页上能做通常控件做不到的事，单击它时，它在两种状态之间切换**

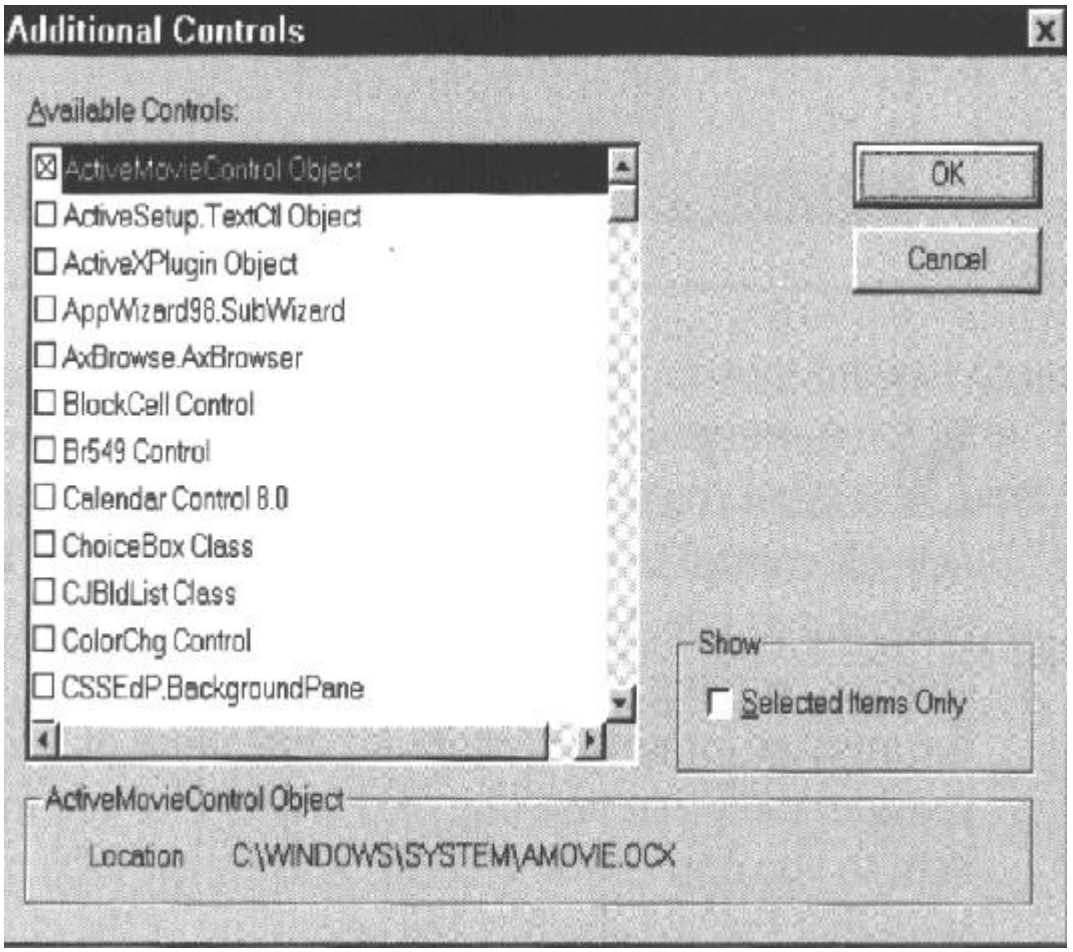
注释 也许你会看到 Satety Violation （安全被破坏）对话框，它告诉你本页的 ActiveX 控件含有不安全代码。这是因为我们还没有把它注册(mark)为安全的。在第 14 章中会学到关于安全性的知识，第 14 章中包括了一旦控件完全测试完成后，怎样把它注册为安全的，现在暂且不用理会这条消息。4.2 版以上的 Microsoft Visual C++ 提供的工具负责处理这些细节（至少提供一个菜单使你不必回到 DOS 提示符下）。本章开头我就说过，可以用旧编译器编译 ActiveX 控件，但新的编译器使得做这类事更方便，上面的情形就是其中的一个方面。

现在，已在 C++和本地 ActiveX 文档中测试了控件的基本功能。在移到网络上去之前，应该试试控件的各种变换，比如，OnOff 控件应能在其它几个标准配置中工作，试一下看看它们是否真能工作。

现在要把 OnOff 控件与其它控件集成在一起使用了。我们在 ActiveX Control Pad 中新建一个 HTML 页。我们先创建一个新的 HTML 框架，然后把 OnOff 控件放入这个框架。使用 File（文件）|New HTML Layout（新建 HTML 框架）命令，创建一个如图 10.10 所示的窗体（这个图还显示了在下面段落中要添加的 On/Off 控件）。就是在这里，你可以拖几个组件来与 OnOff 控件集成在一起。注意到示例框架中为了进行测试，只把几个控件放到了窗体上。在这时，确实不必添加太多的复杂性，只要有足够的信息来保证控件能在一个标准的 HTML 环境中工作就可以了。

注 有趣的是，我们在工具箱中看到的所有控件，实际上都是 ActiveX 控件而不是标准 HTML 控件。

这次添加 ActiveX 控件的过程与以前有些不同。右击 Toolbox（工具箱），就会看到上下文菜单，选择 Additional Controls （附加控件）选项，就会看到如下图所示的 Additional Controls（附加控件）对话框。



Additional Controls（附加控件）对话框包含了你的机器上全部 ActiveX 控件的完整列表（实际上不用管这些控件会不会是 OCX）。还能够选择显示出可插入对象的完整列表。例如，你可以把 Microsoft Word 文档或一个 Excel 电子表格放入窗体。在第 11 章我们将讨论这个特殊过程。

从列表中选中 OCXExmpl 可选项（或者是你命名的控件名称）。要保证确实把挨着这个控件的复选框选中，否则，ActiveX Control Pad 不安装它。单击 OK 则大功告成。你会看到一个新控件添加到了你的 Toolbox（工具箱）中。抓取住它并把它挨着 Command（命令）按钮放下，如图 10.10 所示。和以前一样，如果使用本章中的示例控件，就要保证把它的 OnOff 属性设置成 True。

完成编辑后，保存你的框架，然后关闭本页。ActiveX Control Pad 将向 HTML 脚本添加一个<OBJECT>标记，就如同前面为我们测试的 ActiveX Control 页所做的工作那样。当然，这时的不同点是我们看到了一整页控件，而不是仅仅只有一个控件在页上呆着。保存新的测试页，然后再用浏览器打开它看看控件能不能工作。在这种环境下对控件进行测试后，你就可以使用我们刚才建立的两个页，移兵到网络中进行测试了。

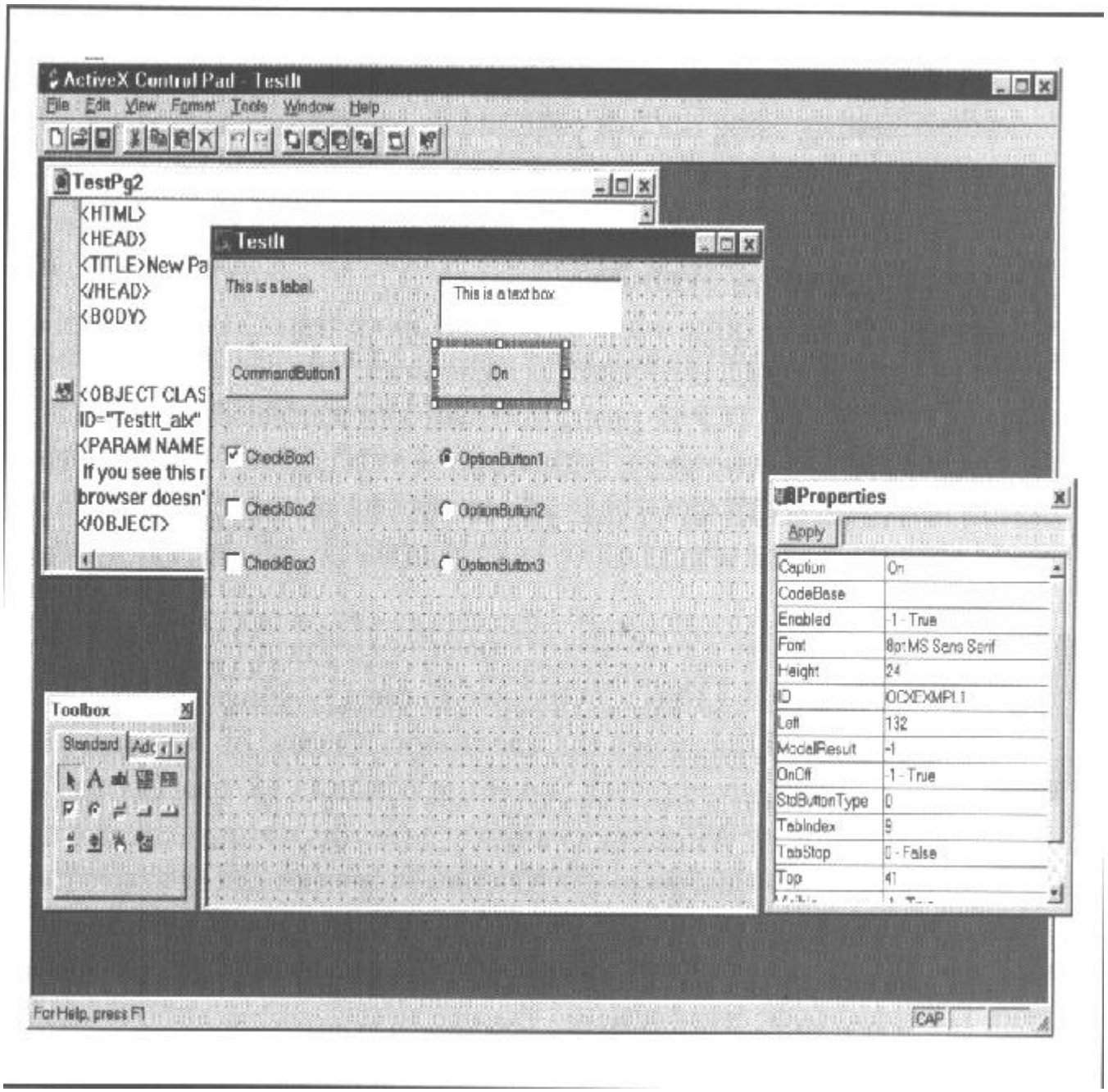


图 10.10　**测试** ActiveX **控件的第二步工作是看一下它能否与其它现存的** HTML **控件一起工作**

内部测试阶段 3： 使用网络连接测试整个 Web 页

在网络上测试控件的方法相当多，唯一的要求是要有两台机器，一个是服务器，一个是客户机，并使用 TCP/IP 协议将它们连接起来。你还可以在服务器上运行 HTTP 服务程序。幸运的是，Windows NT 4.0 的两个版本都自带有小型的 HTTP 服务器，你可以用它进行测试。事实上，Windows NT 的新版本还将包括 FTP 和 Gopher 服务器。

Web 链接 Windows 95 用户进行 Web 页的本地测试时，不用再羡慕 Windows NT 邻居了。现在也可以得到 Windows 95 的个人 Web 服务器了。Windows95 的 OSR2 版的拥有者已把这一特色作为软件包的一部分使用了，这只需打开 Network Properties（网络属性）对话框，并安装它就可以了。如果你的 Windows95 是老版本，你可以从 http://www.microsoft.com/msdownload/ieplatform/iewin95/10000.htm 下载所需的支持。在 http://www.microsoft.com/windows/pr/win95osr.htm 可以发现为 Windows95 可以下载哪些 OSR 功能。

你要保证能用标准浏览器和你指定的域名与服务器通信。不要错误地认为，可以通过像浏览器（如 Explorer）这样的东西访问服务器，然后双击测试页。不错，这将打开浏览器，你将看到你的测试页。但是，你将使这一测试阶段的目的遭受挫折！通过 Explorer 或文件管理器(File Manager)打开这个页，将使浏览器置于文件模式，它不允许你测试 HTTP 服务程序的能力。这一测试阶段的

目的是，尽可能紧密地模拟典型用户访问你的站点时的环境。

一旦与服务器建立了链接，就能看到它提供的缺省的 Web 页。在其中添加一个到你的测试页的链接，使用这个链接就能测试前一节中创建的单个控件和整个 HTML 页。多数情况下，你会发现，对于测试的这个阶段而言，简单地测试一下链接就很充分了。换言之，如果控件在一种模式下能工作，那么，它就应该在你提供给它的全部其它模式下也能工作。例如，在我们的示例控件中，我把它作为一个 on/off 开关进行简单的测试，那么，标准按钮模式也应工作得很好，因为，在测试的前两个阶段，我已经彻底地测试过它们了。这正是本地测试之所以重要的原因，它能在这个阶段节省时间。（想象一下，不得不反复创建同一 Web 页的几个版本，并在服务器连接上测试它们，对于程序说来，这是多么没有效率的方法呀！）

外部测试：在未被污染的机器上检查控件的功能

在我们的私有环境中，已对控件进行了彻底的测试。考察了每种控件访问，我们确信控件可以与标准的 HTML 控件一起工作。现在该进入外部测试阶段了。

实际上，这一阶段仅仅是第三个内部阶段的扩充而已。但是，不再是从网络上访问 Web 站点，而是像你的用户那样，通过电话连接来访问站点。另外，你对控件进行测试一直使用的是受了污染的机器，在这种机器上，已安装了被测试控件，而且，拥有了所有适当的注册表项。

这个测试阶段的目的分为两个方面。第一方面，想确信，当从一个未被污染的机器上进行访问时，控件仍能工作；否则，就会发现你的控件依赖于某些注册表项，或者依赖于客户机访问你的 Web 站点时不能满足的其它要求。第二，

想检测你的 Web 页的下载时间，到现在为止，我们一直通过高速连接访问测试页，而使用你的站点的人就没有这种便利了。他们中的大多数人是通过拨号连接来访问你的站点的。

技巧 可能你想把你的测试页安装到你的 Web 站点中一个私有部分，从而防止其它人访问他们。与网络测试阶段不一样，不要在你的主 Web 页上放上对测试页的链接，因为你已经在以前的测试阶段中测试过连接性能，所以可以用直接地址访问测试页。

为了完成这一部分的测试工作，把测试页移到你的 Web 服务器的一个私有部分。从一个未被污染的机器上，使用一个支持 ActiveX 的浏览器，拨号进入你的 Web 服务器。你需要使用你的网络的私有部分的 URL 而不是主页的 URL 。先试一试单一控件测试页，看看 ActiveX 控件是否能工作，还要看一看在没有一个整页实现的影响时下载控件用多长时间。一旦检测完单一控件测试页，就转向使用整个 HTML 框架的第二页。若这个连接成功，你就获得了一个可以工作的 ActiveX 控件，这个控件能够用于增强你的正规 HTML 页面的功能。

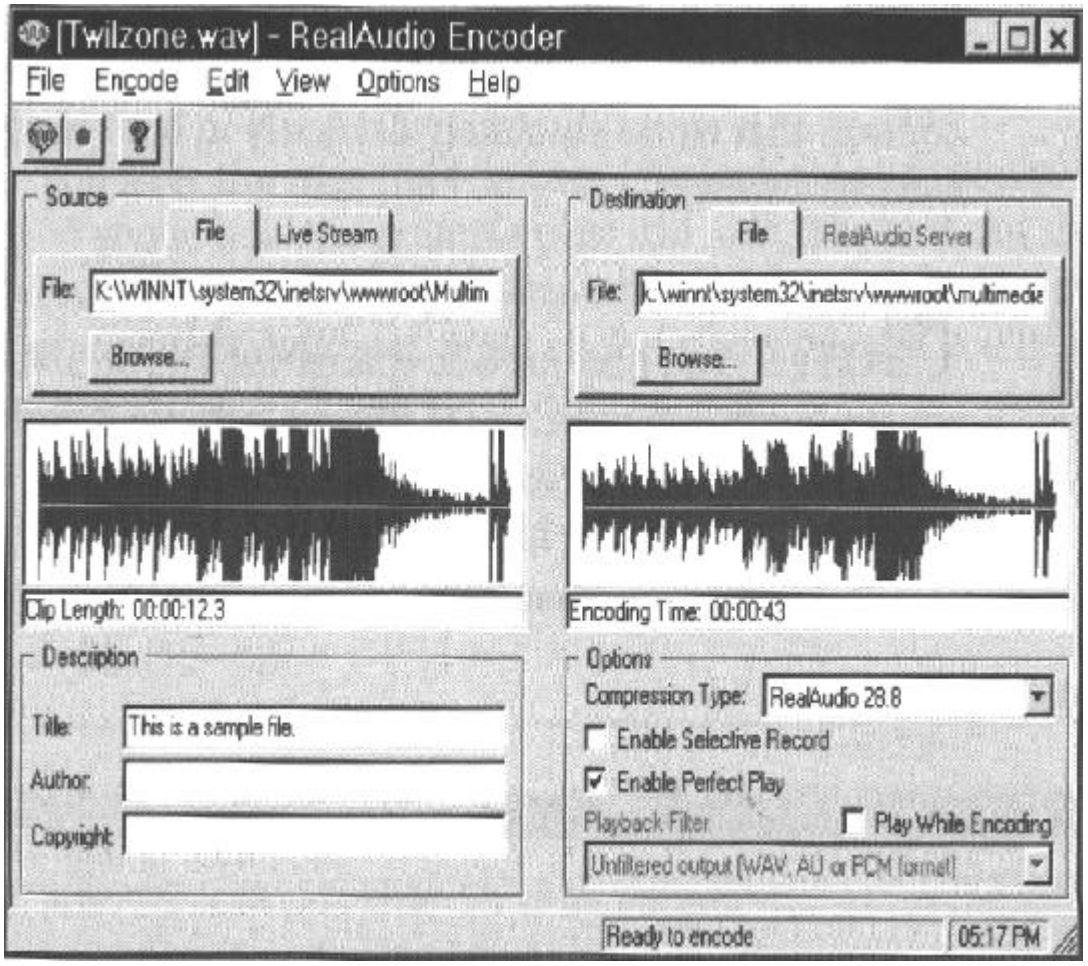
10.3 在 Netscape Navigator 和 Internet Explorer 中使用 ActiveX 控件

演奏音乐以及其它形式的音响效果，大概是使 Web 页更具吸引力的最普通的方法。RealAudio 在提高你获得的声音质量时，至少使得这个过程容易些。

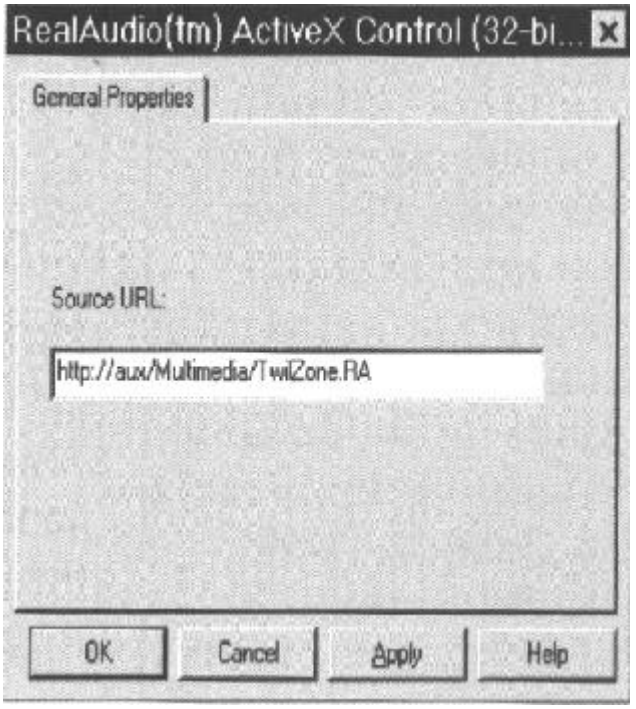
RealAudio 有好几种不同形式，而不仅仅是一种 ActiveX 控件。这个多媒体的当前版本还可以做为 Netscape Navigator 的插件。在 http://www.realaudio.com/可以与 RealAudio 的制作者联系，从而了解播放器、SDK 和支持的实用程序的详细情况。SDK、RealAudio Encoder（编码器）、RealAudio TimeLine Editor（在线编辑器）、RealAudio Personal Sever（个人服务器）以及 RealAudio Content Creation Guide（内容创建指南）是可以免费下载的。RealAudio Personal Sever 支持两种数据流——这对于用于演示和试验目的来说是太多了点。

这个播放器提供的多层次支持，只是使用这一特定技术的原因之一。相当多的站点都使用了它。最有趣的站点是 http: //www. cdnow. com/ （它与 Magazine Warehouse 相连接）。这是一个使用多媒体效果的 CD/movie（电影）商店。例如，如果你打算买一张 CD，那么在购买之前可以先在你的 PC 扬声器上听一听所选的乐曲片段。它还卖 T 恤衫，你可以在购买之前先看一看货。唯一做不到的是不能在购买电影之前先看一些电影片段，但几乎可以肯定的是它最终会添加这一功能。

启动 RealAudio 不成问题，要做的只不过是安装服务程序，对你的文件编码，然后安装 Web 页。RealAudio Encoder 实用程序对话框如下图所示。只需选择将要编码的文件，在 Description（说明）区键入所需的信息，然后单击 Encode 按钮对文件编码（RealAudio Encoder 实用程序自动地为你指定一个目的文件名）。



把这个控件添加到Web **页也相当容易。在**ActiveX Control Pad **的**Insert ActiveX Control**（插入**ActiveX **控件）对话框的控件列表中，选择**RealAudio **控件——以前我们所做的也是这样一个过程。除了**CodeBase **这一个属性需要改变外，其它需要改变的属性如下图所示，把声音文件所在的**URL **打入即可。**



程序列表 10.7 **是这个示例的代码。注意它包含了** NCompass ScriptActive **所需的**<EMBED>**标记。它是在** Netscape Navigator **和** Internet Explorer **上都能无毛病地进行工作的少数控件之一。**

注释　也许需要改变一下 RealAudio1<OBJECT>**标记中** HEIGHT **和** WIDTH **属性，使它能在你的特定的显示配置中工作。当前值代表的是缺省尺寸，也许有些人会发现在高分辨率显示中它太小了。**

程序列表 10.7

```
<HTML>
<HEAD>
<TITLE> RealAudio Test </TITLE>
```

```
</HEAD>
<BODY>

<!-显示标题->
<CENTER><H2>Test of RealAudio ActiveX Control </H2></CENTER>

Click here to test the control:<BR>
<!-显示控件->
<OBJECT ID="RealAudio1" WIDTH=49 HEIGHT=39
        CLASSID="CLSID:CFCDAA03-8BE4-11CF-B84B-0020AFBBCCFA"
        CODEBASE="http://aux/controls">
        <PARAM NAME="_ExtentX" VALUE="1296">
        <PARAM NAME="_ExtentY" VALUE="1032">
        <PARAM NAME="SRC" VALUE="http://aux/multimedia/TwilZone.RA">
        <PARAM NAME="AUTOSTART" VALUE="0">
        <PARAM NAME="NOLABELS" VALUE="0">
<EMBED NAME="RealAudio1" WIDTH=49 HEIGHT=39
        CLASSID="CLSID:CFCDAA03-8BE4-11CF-B84B-0020AFBBCCFA"
        CODEBASE="http://aux/controls">
        TYPE="application/oleobject"
        PARAM__ExtentX="1296">
```

PARAM__ExtentY="1032">

PARAM__SRC="http://aux/multimedia/TwilZone.RA">

PARAM__AUTOSTART="0">

PARAM__NOLABELS="0">

</OBJECT>

</BODY>

</HTML>

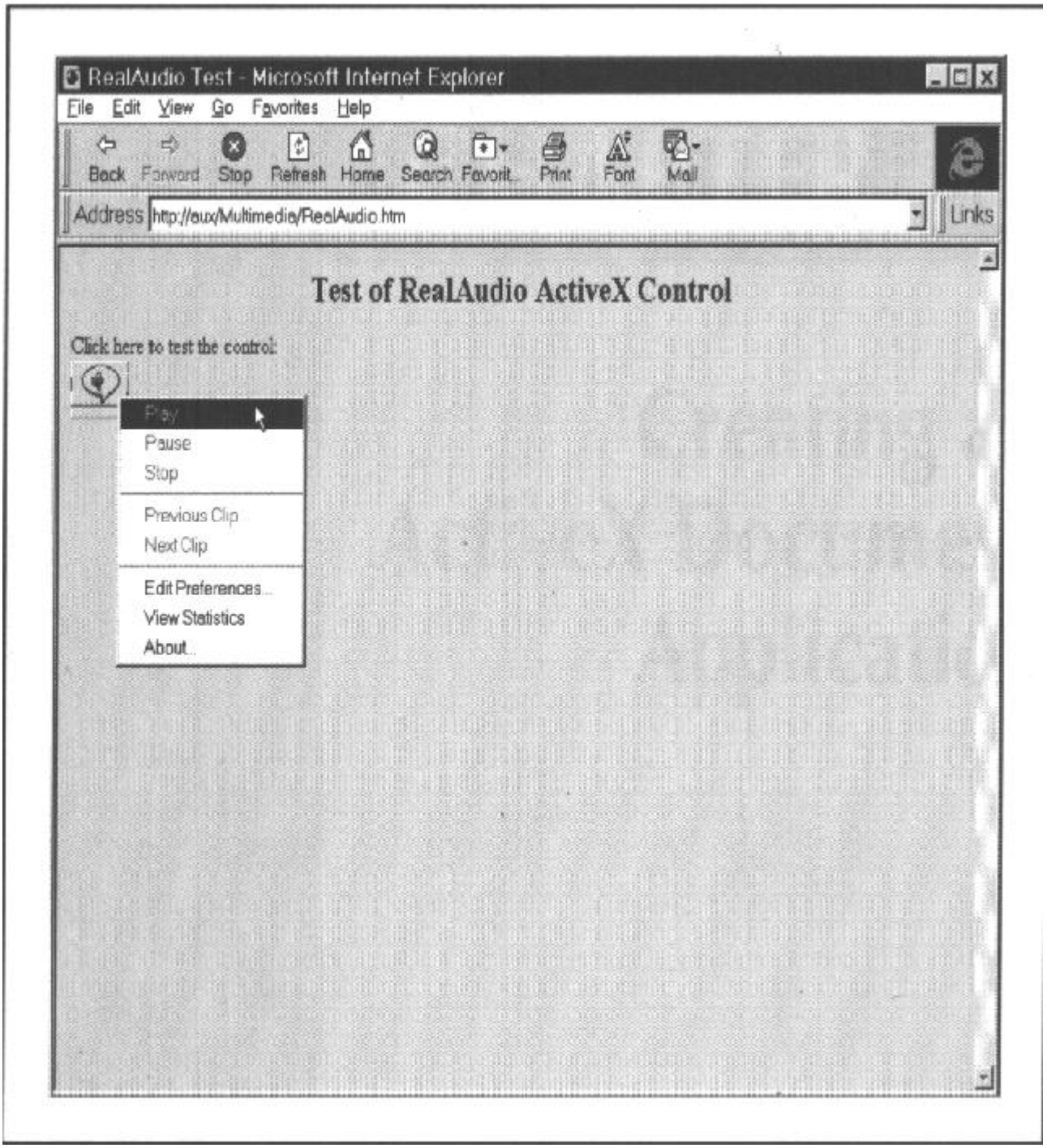
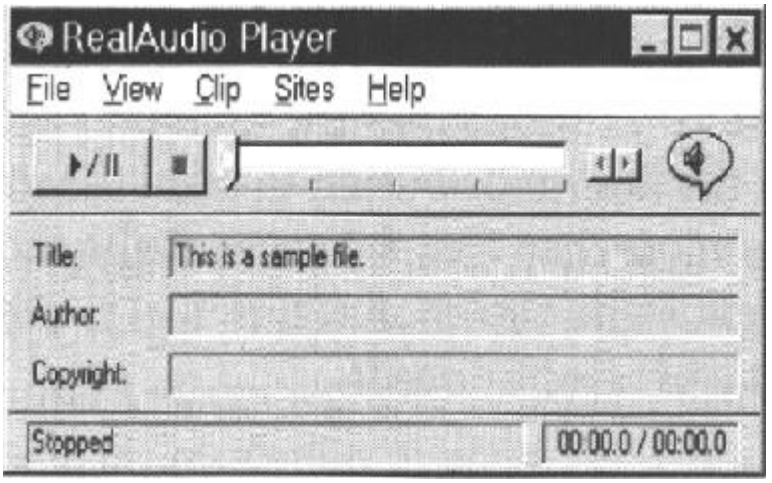


图 10.11　RealAudio **控件提供了独特的按钮接口以及关联菜单供用户进行设置**

现在把代码写好了。来看一下最后结果吧。图 10.11 显示的是这个例页的外观。用户只要单击 RealAudio 按钮就可以打开声音文件。这个控件还有别的功能。右击该控件，就会看到如图 10.11 所示的上下文菜单，用户可以定制控件以满足专门需要。

那么，该控件操作起来是什么样子呢？下图显示了用户眼中示例 WAV 文件的显示情况。



请注意，这个对话框中显示了在 RealAudio Encoder 实用程序中添加的说明信息。遗憾的是，你添加的信息量受到限制。最好除了控件所提供的缺省域外再添加一个用户可配置的域。

第11章　创建ActiveX文档应用程序

到目前为止，我们已讨论了各种新技术，包括 ActiveX 和 COM+，这些技术允诺要把 Internet 变成每个人的业务工具。使用 ActiveX 的方式多种多样（COM+ 在将来会用到）。例如，Microsoft 发布了一个称之为 ActiveX Accessibility 的 ActiveX API 。这个 API 设计用于向 Internet 添加那些已添加到 Windows 95 中的Accessibility applet(含有内容的小程序)。类似于Sticky Keys（一种每次按一个键而不是同时按住所有键的方式来创建 CTRL 组合键的方法）这样的方法，不仅出现于 Windows 95 和 Windows NT 4.0 桌面系统，而且正在出现于与 ActiveX 兼容的 Internet 浏览器中。

还有其它一些（由 Microsoft 和其它公司开发的）基于 ActiveX 的技术已经投入使用——这样的技术实在太多了，这里难以一一罗列。更重要的 ActiveX 技术之一是 ActiveMovie，事实上，MSNBC Internet 站点已经使用 ActiveMovie 提供 NBC 新闻网络的新闻剪辑了。实际上，ActiveMovie 使你能在 Internet 上看电影剪辑、放映 AVI 和其它种类的电影文件。

注　ActiveX 提供了把令人激动的新技术从桌面系统转移到 Internet 的手段。

WEB 链接 刚才讨论的新技术可以查到更多的可用信息。可以在 http://www.microsoft.com/enable/dev/msdn4.htm 中找到 ActiveX Accessibility。ActiveMovie 控件的使用说明和技术手册出现在 http://www.microsoft.com/msdn/sdk/inetsdk/help/complib/activemovie.htm 中。

最重要的 ActiveX 技术并不是新颖而令人激动的技术，而是涉及到一个老生常谈的问题，即在 Internet 上共享文档。因为接口是静态的，这确实一直是一个大问题。读过第 8 章后，考虑一下为了创建任意种类的动态接口所需的程序复杂度和所需的标记数量，就会认识到这一点。是的，你能把字处理文档结合一些现实的数据嵌入到 HTML，但是数据并未变化，用户还是难于编辑它。对于电子表格也是一样。显示数据一般说来并不太难，只要你肯于牺牲最新信息。ActiveX 提供了解决办法，它允许你创建动态文档，你可以编辑它并且看到它的实时变化。

那么，这一技术是从什么地方神秘地冒出来的呢？其实它一点也不新颖，Microsoft 只不过简单地修改了 Microsoft Office 中的技术而已。ActiveX Document 原来的名称为 OLE Document Objects（文档对象）（大多数人，包括 Microsoft 的人，已把 ActiveX Document 简记为 ActiveDocument，本书中我们将一直使用这个术语，原因在于它既清楚又简单）。它是 Microsoft Office Binder 技术的一部分，从未打算成为一个公开的规范。最初甚至在能得到 OLE Document Objects 规范之前，必须申请 Office 兼容的程序。只是到了 Windows 95 出现之

后，这一规范才成为每个人都能得到的东西。有迹象表明，Microsoft 会把这一规范公开的，原因在于它是 OLE 演变中的下一个逻辑步骤。

本章的中心内容是 ActiveDocument。我们要考察一下，除了数据应用之外，什么是 ActiveX 的最重要的用途。另外你还将看到，创建浏览器使用的 ActiveXDocument 是多么容易。另外，本章还将介绍一下创建你自己的定制文档的一些设置时，你需要做些什么工作。

注 你可能会听到有人将 ActiveX 文档称为 OLE 文档对象，另外有些人将它们称为 ActiveDocument 文档。

本章将介绍三个主要的 ActiveDocument 主题。首先让我们看一下 Web Publishing Wizard。如果你不了解从用户角度看 ActiveDocument 是如何工作的，那么你应该密切关注这一节的内容了。总而言之，怎么能期望写出你自己都不理解的应用程序呢? 接下来我们考察一下 ActiveDocument 的理论基础，最重要的是你要实现的接口。最后，我们创建一个简单程序，它实现了 ActiveDocument 所需的接口。

注释 本章的例子全部采用 Visual C++ 6.0，但是，除了一些小的程序性变动之外，也可以容易地使用 4.2 以上版本的 Visual C++（代码本身应该工作的一样好）。你必须拥有 Visual C++ 的某种版本（推荐 6.0 版），以便于从头至尾地学习本章的例子。也可用 Visual C++4.1 来学习这些例子，但你在工作中却要尽量少用为好。例如，可能需要添加对 Visual C++4.2 以上版本作为缺省支持的各种新类的引用。另外，创建例子时，也不如使用新一些的产品那样，自动化程度高一些。

理论上，使用 Visual C++4.0 创建这些例子也可以。但所需的工作量很大，无疑使用该编译器的升级版显得更容易些。

11.1 什么是 ActiveDocument（OLE 文档对象）

到现在为止，对于用户在 Internet 比较熟悉的文档种类，我们还没有进行太多的讨论。例如，用户非常熟悉刚在本地硬盘上修改过的 Word 文档，但他们在 Internet 上使用真正干练的 ActiveX 控件却是不完全熟悉的。对于使用只有程序员喜欢的方式来显示专用页元素的脚本语言，我们也要讨论（用户大概不会全都对创建 JavaScript 的最新技术感兴趣）。当然，这就会带来差不多每个人都会问到的问题：对于用户而言，除了一点点信息、几个窗体以及那眼花缭乱的 ActiveX 控件，Internet 到底是个什么玩艺儿? 那么，ActiveDocument 为你提供一个答案，它们是对每个人说来都可用的在 Web 页上创建内容并显示数据的一种手段。

但是，ActiveDocument 却远远不止是共享信息的一种方式，你可以使用它们正儿八经的干点事。考虑一下这种场景吧：你的公司拥有自己的内部网，公司的雇员们需要时可以在世界上任何地方进入这个网，他们使用这个网获得自己的 e-mail 并完成各种各样的其它任务。现在你的老板告诉你，要求市场开发部提交报告，而这时大部分销售代表都在外工作着，你怎么样才能完成这一工作呢?

注释 Netscape Navigator 和早期版本的 Internet Explorer，通过使用帮手(helpers)让你看到非标准文档。浏览器启动应用程序的一个完整拷贝，并把 Web 站点上的文件内容传递给它。这种办法的问题是，启动另一个应用会使用更多的内存，而且你与 Internet 服务器也没有现场连接，所以对文件所作的改变不会立即反映到服务器中去。尽管 Microsoft 在 Internet Explorer3.0 及以上版本中仍然使用“帮手”(helpers)一词，但从用户的观点来看两者的意义却是完全不同的。

在这种情况下，ActiveDocument 就是答案。使用第 8 章中讨论过的标记，在 HTML 文档中建立到该文档的链接。当用户单击文档引用时，他们在浏览器中看到的是文档的可编辑版本，这个浏览器的菜单和工具栏发生了一些变化。图 11.1 中显示的是在 Internet Explorer 浏览器窗口中显示的 Word for Windows 文档。请注意浏览器所有特色仍然完整无损，变化了的只是文档显示的方式。这种特定技术并不是什么新货色，它被称之为在位编辑(in-place editing)。大多数 OLE2 服务器现在都能向 OLE2 客户机提供这种功能，你可以试一下 Word for Windows 或 CoreDRAW！单击 OLE 对象时，菜单和工具栏会随时改变得与客户机上的菜单和工具栏匹配。最大的差别在于，这是跨越 Internet 连接在浏览器中正在发生的事。

注 ActiveDocument 使用称之为在位(inplace)编辑的 OLE2 特色，它让你在客户端中编辑文档而不是作为一个独立的服务器进行编辑。

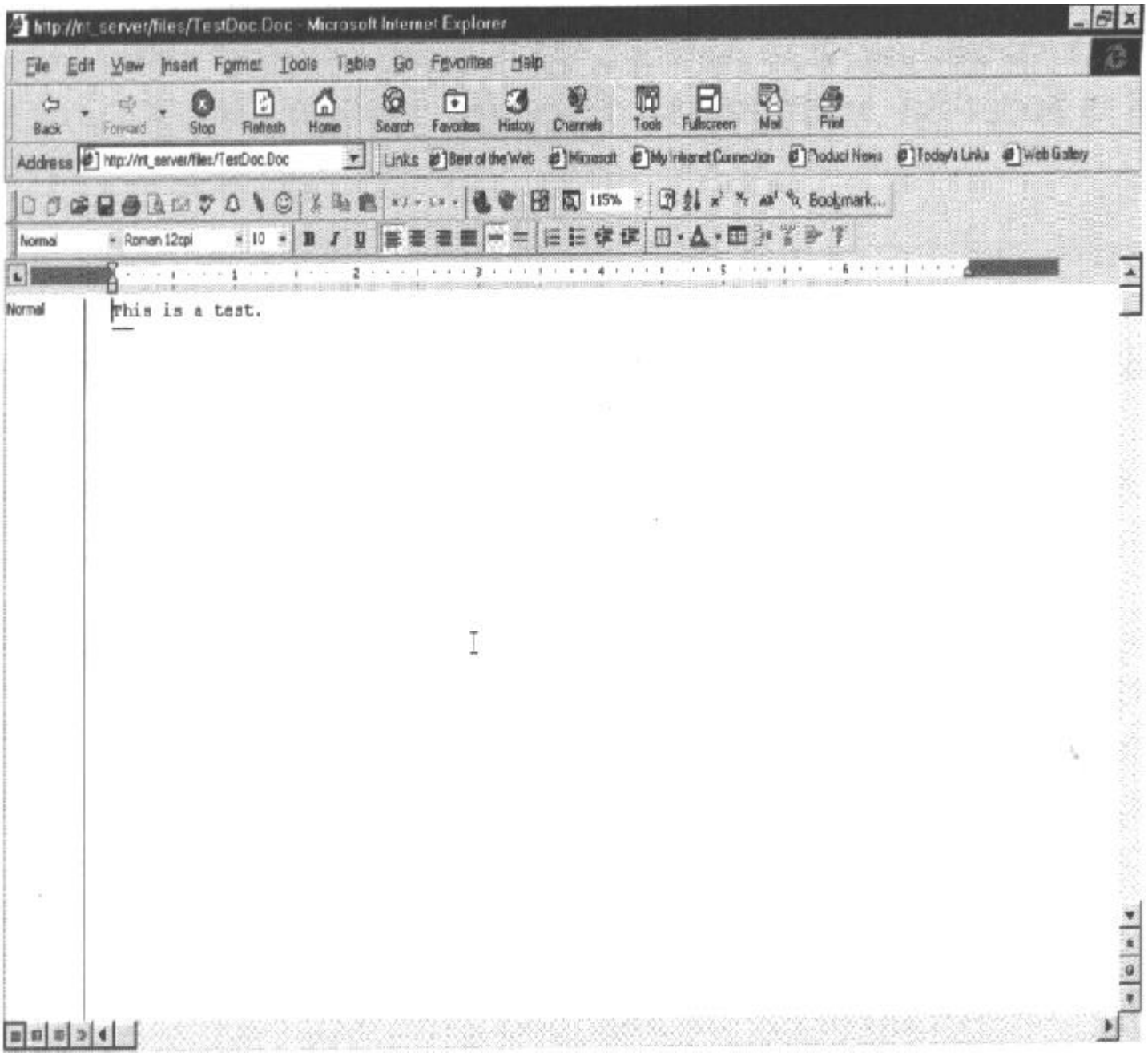
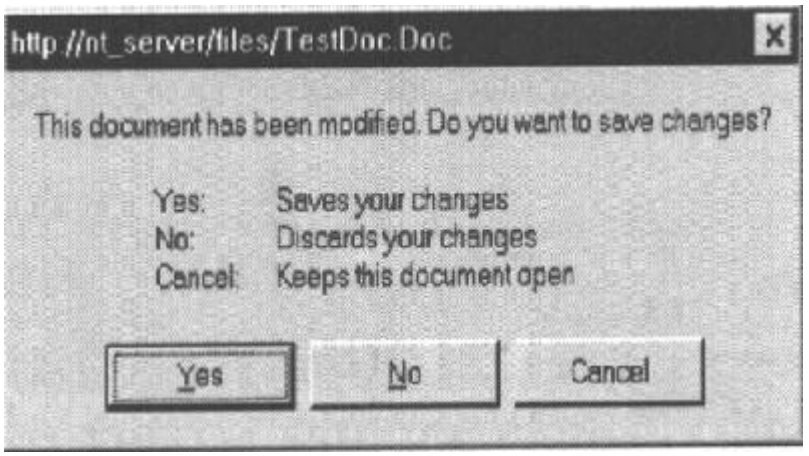


图 11.1　Internet Explorer 4.0 **显示了在位**(in-place)**编辑的效果，这是** COM **的** OLE2 **特色**

注释 如果在基于 LAN 的 Internet 服务器上测试这个示例的话，可能会出现一个对话框，问你是否想打开或保存文件。多数情况下，在一个现场(live)Internet 连接的连接中测试时，肯定会出现这个对话框（取决于你是如何设置浏览器安全性的）。简单打开文件，就会看到如图 11.1 所示的效果。

你还会注意到另外一个差别。如果改变了这个文本的内容，然后在 Internet Explorer 浏览器工具栏上单击 Back 按钮，就会看到如下图所示的对话框，它询问你是否想改变文件。如果单击 Yes，就会看到典型的 File Save（保存文件）对话框。遗憾的是，使用这一技术保存文件将把文件保存到你的本地硬盘中——在我们这里这项特性没有什么用处，但也许在别的地方有用。还有其它保存文件的方法，可以使用 File（文件）|Save （另存为）命令，你还能够使用 File（文件）|Send To（发往）| Web Publishing Wizard（发布向导）命令，真正地将文件送回 Web 服务器。这正是我们所要的。（Internet Explorer 与 Windows 的某种组合不提供 File|Send To|Web Publishing Wizard 命令，这就是说，必须先把文件存在本地硬盘，然后再用 Web Publishing Wizard 命令将它发往 Web 服务器）。本章后面一节‘使用 Web Publishing Wizard ’中，将讨论使用 Web Publishing Wizard 所需的步骤。



技巧 对某些情况来说，本节讨论的每种技术的帮助都不太大。例如，你可能想向 Web 服务器邮寄文档，而且希望每个邮件单独到达。这时，你会告诉用户使用 File|SendTo|Mail Recipient（邮件收件人）命令。显然，这就意味着你需要整理所有收到的文档，但是，当其它方法无效时，这种方法却能够使用。

技巧 当前的HTTP1.0规范不允许用户向WWW服务器发布文档，新的HTTP1.1规范修正了这一点。如果你希望允许用户从 Web 页面上向服务器发布文档，就应提供一个使用 1.1 规范的服务器。因为不必编写通常的脚本或者在连接中跳来跳去，所以你不仅可以节省时间和精力，而且，可以减少电话支持数量，从而对用户而言事情也就更简单了。（Netscape 和 Microsoft 产品的所有当前版本都支持该 1.1 规范，如果你拥有任一家供货商产品的老版本，可以查看一下自己的文档。）

ActiveXDocument 的变化也使得程序员必须考虑对 OLE 的支持。在客户机有三级对象参与。对象可以简单地出现在查看区，也可以占据整个窗口，还可以占据整个应用框架。现在来比较一下这三级对象的不同之处。

以前，你可以创建一个对象并将它放在容器中。OLE1 出现后，容器只是简单地显示一个图标，说明对象的存在。如果希望编辑对象，可以双击容器中的对象图标。Windows 将在另一个窗口中形成该文档的完整拷贝。这正是某些人所说的对象简单地出现在浏览区的意思。

OLE2 改变了客户机和服务器交互的方式。现在，你可以真正看到对象的内容了。例如，如果在字处理文档中放入一个电子表格对象，不必双击该对象就能看到它的内容。它就是窗口级的参与者。客户机和服务器共享窗口；客户机显示它的数据，然后依靠服务器显示对象内部信息。OLE2 还提供了在位替换操作。大部分情况下，双击对象都会启动一个进程之外的(out-of-process)服务程序，它占据整个客户机框架。服务器占据了一般留作客户机使用的菜单和工具栏。从用户角度来看，应用程序是一样的；只是改变了工具，以满足用户编辑对象的需求。

注　当你创建了一个与外部文档的连接时，OLE1 只是显示一个图标；OLE2 则显示真正的文档内容并允许进行在位编辑。

通过使用浏览器，ActiveDocument 将该技术从桌面系统扩展到了 Internet 上。现在浏览器工具将会自动更新，以适应用户的需要，用户不必再打开一个其它的程序来编辑文档。一个进程之外的服务程序将负责更新浏览器菜单和工具栏，以适应应用程序的一般需要。实际上，Windows 的未来版本将会做得更好。

注　当你单击窗口应用程序中的对象时，进程外的服务程序实现你看到的菜单和工具栏的修改。

Windows 95 和 Windows NT 4.0 使用的都是资源管理器式界面。如果你用右键单击与某个已注册的应用程序相关的文档，就能看到一个有多种选项的菜单。最常见的选项是打开或打印文档。大部分情况下你能看到使用 Quick View 实用程序浏览文档内容的选项。

用户界面在不久的将来也可能有所改变。双击文档能进行在位操作。服务器将真正接管 Explorer 的菜单和工具栏。用户再也不必离开 Explorer 去打开另外一个应用程序窗口。另外，Internet 站点就像硬盘一样出现在 Explorer 中，并组合了目前的浏览器的功能。

程序员们很欢迎这些变化吗？并不全是。有些人已经在抱怨该技术只是加强了 Microsoft 对计算机世界的控制。当然，只有在你不能安装另一个服务器来替代 Internet Explorer 时才会发生这种情况——这几乎是不可能的事。随着计算技术日趋成熟，ActiveDocument 将会成为一种特别重要的技术。这也正是本章为什么重要的原因——本章的目的正是使你跟上这种日新月异的技术的发展，以便你可以使用它来满足目前的计算需求。

创建连接

现在，你已经了解了使用 ActiveDocument 的结果，并讨论了该技术为什么重要的原因，本节将介绍实现它的 HTML 代码。程序列表 11.1 显示了用于创建该示例的代码，这个示例虽然很短但很实用。它没有增加任何特别功能，从而

使你可以看到创建 ActiveX 文档链接的最小需求。可以看出，代码中只是使用了一个简单的链接。所有该应用程序后面的“魔法”都存在于 Internet Explorer 中。第 8 章已经用这种链接来显示 Web 站点上的其它页了。

程序列表 11.1

```
<HTML>
<HEAD>
<TITLE>ActiveX Document</TITLE>
<BODY>
<CENTER>
<H2>ActiveX Document Test Page<H2>
<EM>Requires Microsoft Word or WordPad</EM><P>
</Center>
<A HREF ="http://nt_server/files/TestDoc.Doc">Test Document</A>
</BODY>
</HTML>
```

好了，现在浏览器中已经有了一个可以编辑的文档了。这确实不是一件特别困难的事情。由于文档仍在浏览器中。所以可以节省内存。只有一个应用程序处于运行状态，所以尽管实际中要花费一些额外的处理时间和内存来浏览及处理文档，但比起运行两个应用程序来说，这样做的花费要少得多。例如，在位激活功能这一特色就是使用进程外服务程序的结果。进程外服务程序实际上

是 DLL 的一种“漂亮”的形式，它能提供与客户机应用程序进行通讯的适宜接口。关键在于：DLL 比完整的应用程序占用的内存要少得多，并且也没有理由担心什么显示问题（这是客户机的工作）。本章“ActiveDocument 结构概述”一节将介绍这种 DLL 的一些需求情况。

注 ActiveDocument 使用进程外服务程序（通常是一个 DLL）来实施在位编辑。

还有两种其它的方法可以创建 ActiveDocument 连接，不过本章将不用多少时间来讨论这些方法。Microsoft Web Browser Control 允许你浏览 Internet 上任意类型的文档，包括那些一般与 Internet 并无关系的文档，如 Word for Windows 文档。另外，在这方面还有一些高级的<OBJECT>标记属性能提供很大帮助。你可能要花费一些时间去了解第 8 章中有关<OBJECT>标记的知识，然后就可以到所列的 Web 站点上去下载有关的技术规范。实际上<OBJECT>标记的高级属性目前正处于不断发展变化中：这也正是本章不作介绍的原因。

使用 Web Publishing Wizard

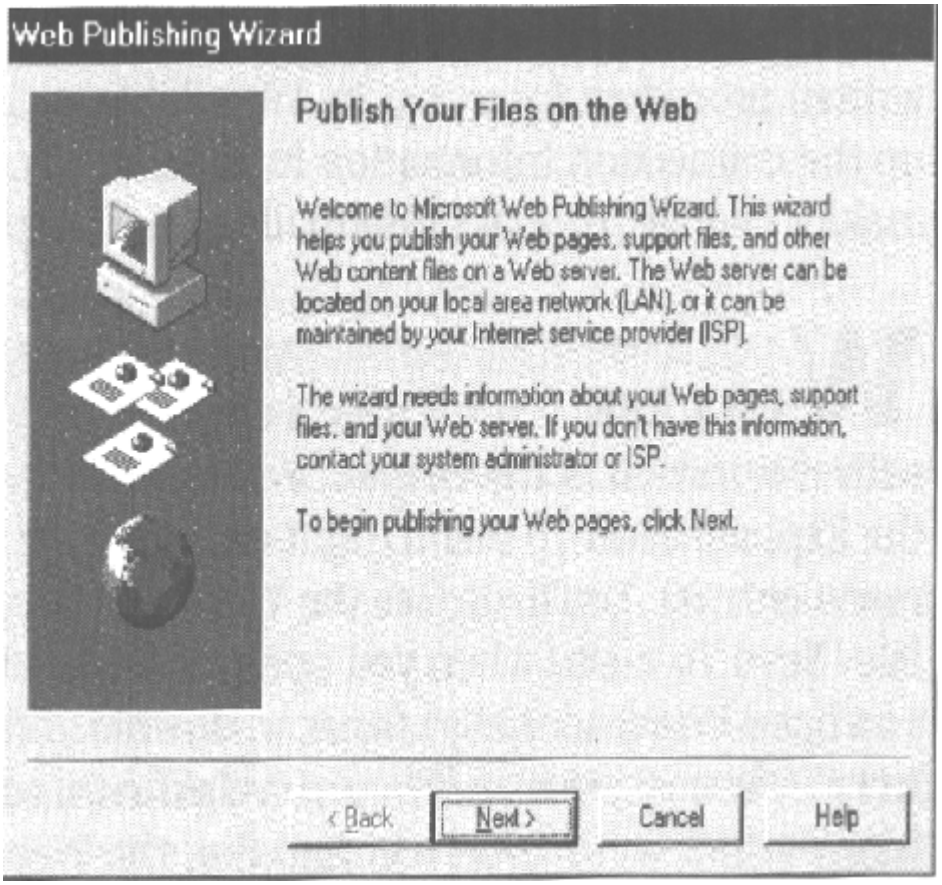
你不必依赖那些老方法来保持文档为最新状态。利用 Web Publishing Wizard，只要稍作努力就能使 Web 服务器上的文档保持最新状态。现在对文档的更改将不会像原来的浏览器技术那样只出现在自己的本地机器上。实际上它们会出现在 Internet 服务器上。由于提供了使用 Web Publishing Wizard （以及可能的高级的设置连接信息）的已经编写好的过程，因此，就可以允许雇员从甚至很远的地方对 HTML 页进行修改了。

Web 链接 学习本章的这一节时，你需要有一份 Web Publishing Wizard 的拷贝。你在“Explorer Send to ”菜单上应该能够看到“Web Publishing Wizard”（右击某文件，然后在上下文相关菜单中查找）。另外，如果你在浏览器中打开一个文档，那么在 Word 的“File|Send To”菜单中也能看到“Web Publishing Wizard”项（但正常打开 Word 时不会看到这个菜单项）。你可以从 http://www.microsoft.com/windows/software/webpost/default.htm 下载 Web Publishing Wizard。Windows 98 用户在添加/删除程序属性对话框的 Windows Setup 标签中，可能会发现 Web Publishing Wizard 出现于 Internet 文件夹中。Web Publishing Wizard 也是作为不同的 Internet 编程工具的一部分由 Microsoft 提供安装的，所以在安装之前应看看 Web Publishing Wizard 是否可用。

那么，怎么开始呢？下面的过程将帮助你进行第一次的实践。第一次尝试之后，你可以将它们总结为简单的四步过程，一旦通过第一阶段后，我们将看一下这个过程的部分内容。我们先从图 11.1 所示的文档起步，并且假定该文档已经编辑过了。现在要做的是将所做的修改保存到 Internet 站点上去。

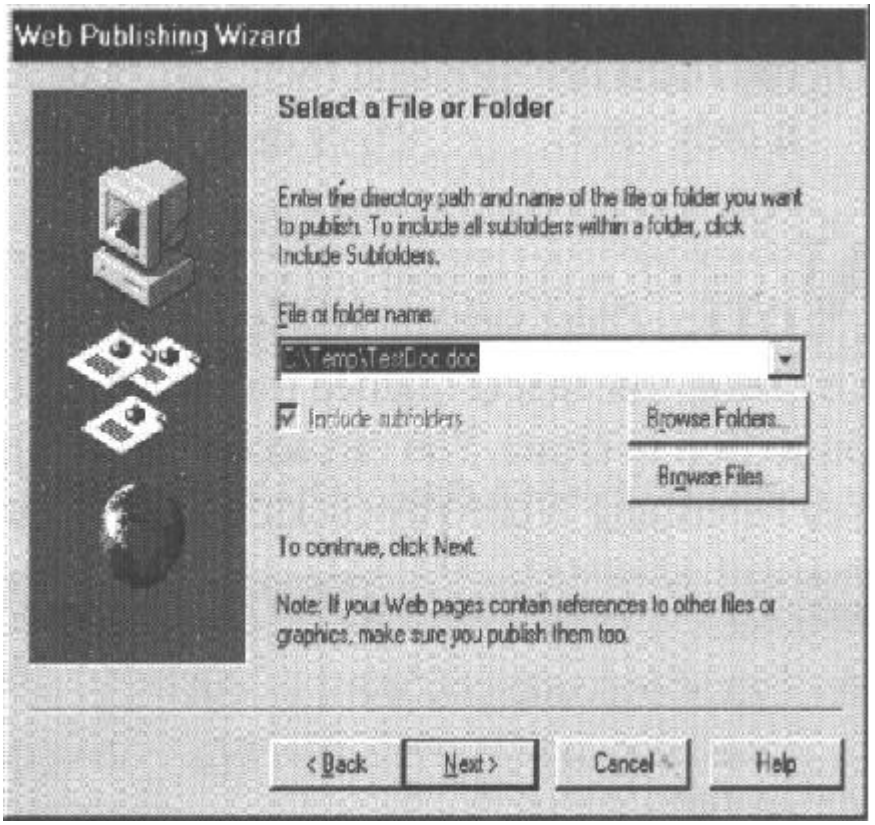
1. 使用“File|Save As ”命令显示“File Save ”对话框。在将文件发送给 Web 服务器之前，必须先在本地机上保存它。可能以后 Microsoft 会修改这个过程，但现在你还是需要花点时间做一个本地拷贝。将文件命名为 Web 站点页的名字。在本例的 Internet 站点中，文件名为 TESTDOC.DOC（如图 11.1 中 Internet Explorer 的标题栏所示）。

2. 在 Word 或 Windows Explorer 中使用 File|Send To|Web Publishing Wizard 命令来显示 Web Publishing Wizard 对话框（如果使用 WindowsExplorer 发送它，要首先选中文件或文件夹）。



3. 单击 Next，如果你使用的是 Web Publishing Wizard 的老版本，或者从开始菜单启动 Web Publishing Wizard，你将看到下一页，如下图所示。在某些情况下，Web Publishing Wizard 将自动把一个缺省的文件名放在 File（文件）或 Folder （文件夹）域中。现在，你还用不到该文件名。（如果你使用的是新版本的产品，那么在启动 Web Publishing Wizard 之前，就要选择一个文件 ，你就会看到一个

对话框，询问你将使用哪一个 Web 服务器连接，这时请跳过第 5 步）。



4. 单击 Browse Files（浏览文件）按钮，系统显示 Browse 对话框（它的外观与标准的 File Open 对话框相似）。找到刚才保存的文档的本地拷贝，然后单击 Open。现在就能在 File（文件）或 Folder（文件夹）域中看到 Web 文档的名字了。单击 Next。

5. 如果你先前曾定义了 Web 服务器连接，就能看到一个包含着连接名称的下拉式列表框。你可以选择其中之一，或者单击 New 按钮，定义一个新的连接。该例中，假定需要一个新连接。（如果确实不需要新连接，可以选择已有的一个连接然后跳到第 14 步）无论你单击了 New 按钮，或是以前从未定义过服务

器连接，都会看到如下图所示的对话框（该对话框来自于 Web Publishing Wizard 的最新版本，它包含有一个描述域和一个命令按钮，而老版本有两个域）。该对话框是你开始定义客户机与 Internet 服务器之间连接的地方。幸运的是，这些工作只要做一次就可以了。

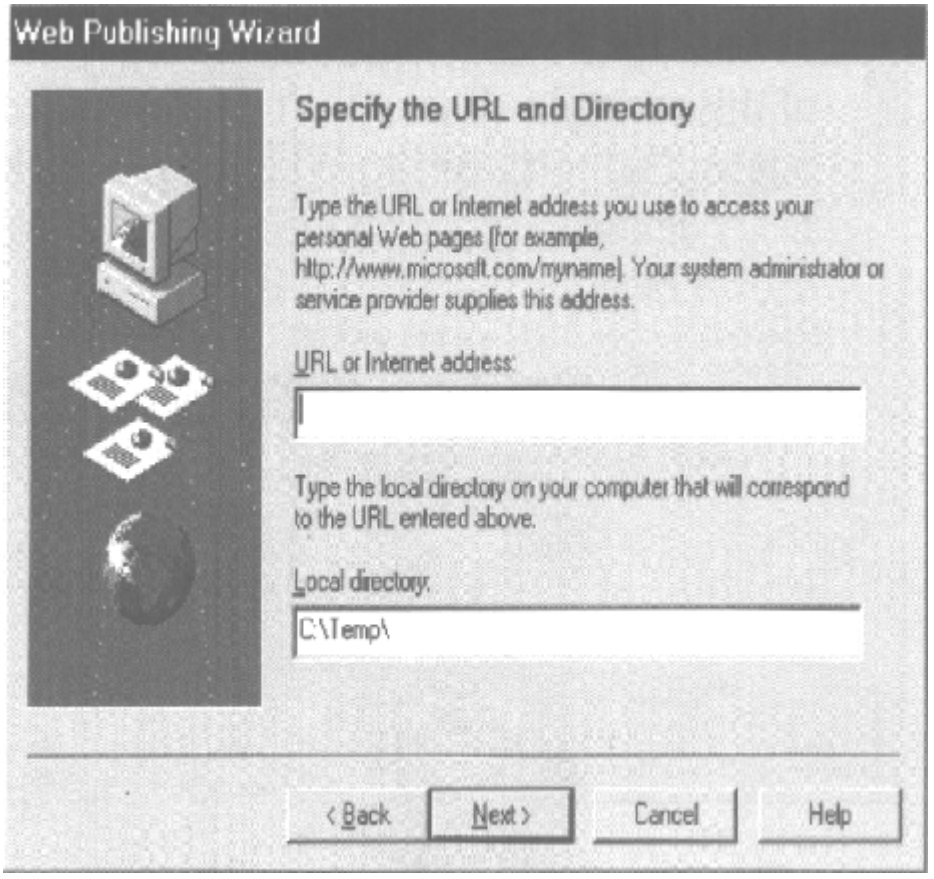


注释 老版本的 Web Publishing Wizard 在这个对话框中提供了两个域，第一个是连接名，第二个是创建的连接类型。可在 Web Publishing Wizard 的新版本上单击对话框中的 Advanced 按钮来访问第二个域。

6. 为 Internet 连接输入一个名字。在绝大多数情况下，如果只需要一项内容的

话，保持 My Web Site 不变就可以了。从第二个列表框中选择 Internet Service Provider（Internet 服务供应者）（Web Publishing Wizard 的最新版的用户需要单击 Advanced 按钮）。如果是为一个 LAN 内部网站点建立连接，那么，当使用老版本的 Web Publishing Wizard 时，可以选择 Other Internet Provider 选项，或者在使用新版的 Web Publishing Wizard 时自动提供 Select Service Provider 选项。

7. 单击 Next，就可以看到下一页面，如下图所示。

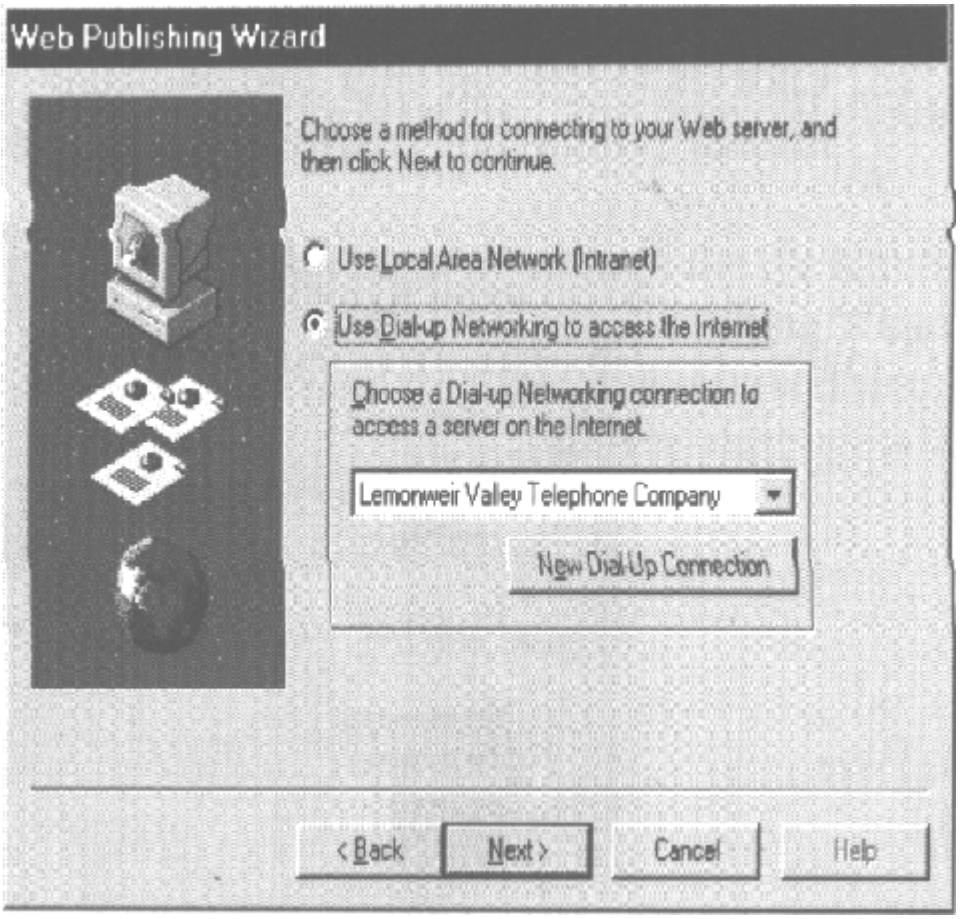


注意，你需要定义一个到 Internet 服务器的连接。在 URL 域中简单地输入自己的站点名。使用 Web Publishing Wizard 新版本的用户还需要提供与远程 Web 站

点相对应的本地目录。这是 Web Publishing Wizard 寻找发送到 Web 服务器上文件的地方。绝大多数情况下使用缺省目录也就可以了。如果你使用 Web Publishing Wizard，那么请跳到第 9 步。

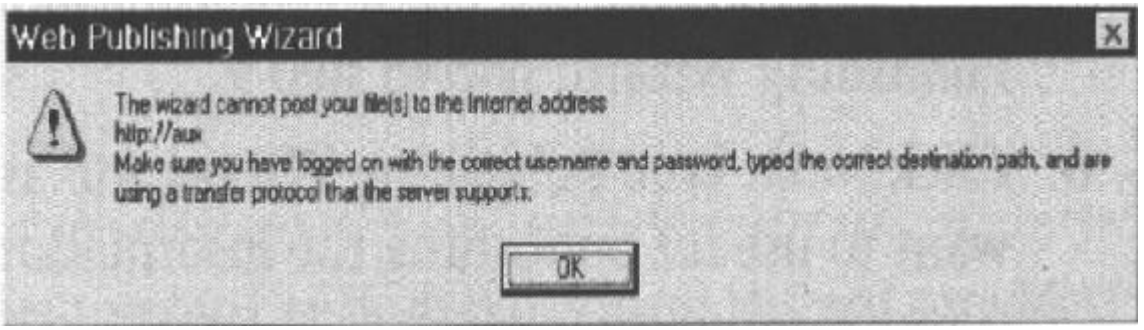
8. 单击 Next，Web Publishing Wizard 就会询问你用于发布信息的 URL，除非使用非正常安装，否则 Web Publishing Wizard 提供的缺省 URL 就工作得很好。

9. 单击 Next，就能看到下一页，如下图所示。该页允许你选择连接的类型：LAN 或拨号方式。不要被对话框中的描述弄糊涂，就像为 Internet 建立连接一样，你可以很容易地为内部网建立拨号连接。

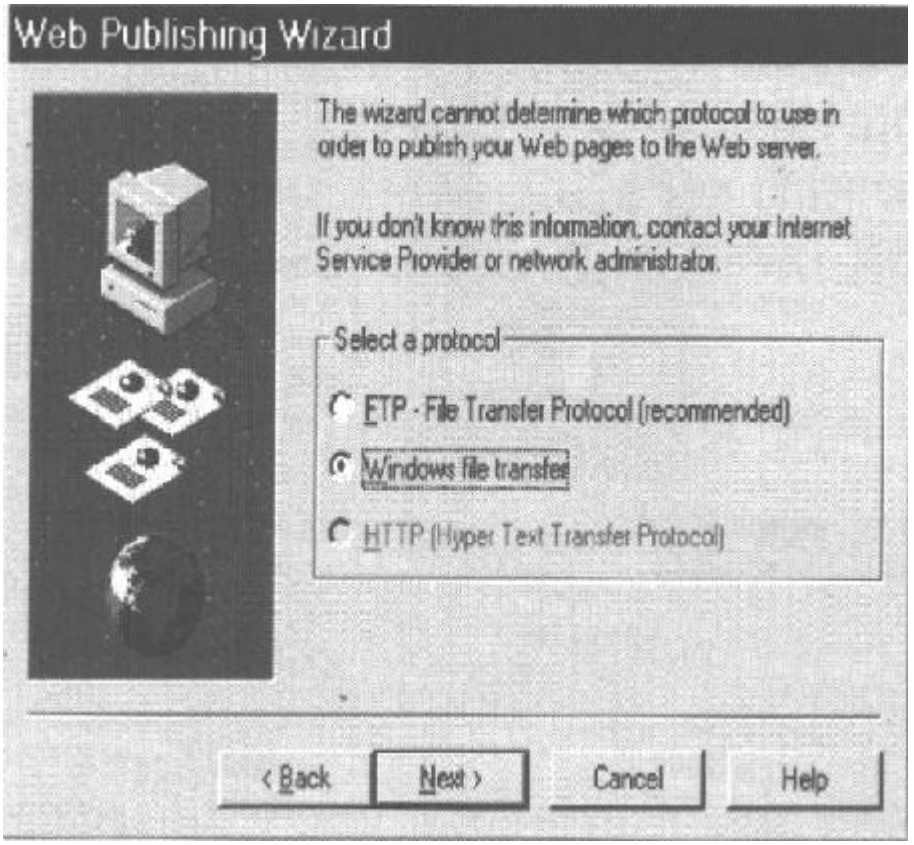


10. 在 LAN 连接（没有调制解调器的连接）或拨号连接之间选择一项。如果你选择的是拨号连接方式，那么还需要在列表框中选择一种拨号连接。单击 New Dial-UP Connection 按钮，可以创建一个新的连接定义。

11. 单击 Next，Web Publishing Wizard 会显示下一页，该页简单说明你所提供的信息中需要验证的信息。单击 Next 将开始对信息进行验证。如果你要建立 LAN 连接，几乎立刻就会得到如下图所示的错误消息。别担心，下面几步会告诉你如何解决这个问题。如果 Web Publishing Wizard 成功地找到了你的站点，你就可以跳转到第 15 步了。



12. 单击 OK，关闭错误消息对话框，你会看到扩展连接配置的第一页，如下图所示。



这就是你选择文件传输方式的地方。如果你是通过拨号连接方式访问 Internet **站点，就可以选择** FTP **或** HTTP **文件传输方式。其中的** HTTP **方法只对使用** HTTP **版本** 1.1 **及以上版本的** Web **站点可用。如果你使用的是** LAN**，那么** FTP **和** Windows File Transfer **选项都可以使用。这里不同的连接还需各种不同的定义。接着，系统将继续显示** Windows File Transfer **选项，这也正是你经常需要使用的选项。**

技巧 如果使用的是 LAN 连接，应尽可能使用 Windows File Transfer 方式，因为它比较快。FTP 连接需要一个附加的文件传输层，这在 LAN 环境中是不需要的。不过，在 WAN 上，FTP 方式能提供附加的安全保护层。

13. 单击 Next，就能看到如图 11.2 所示的下一页。在这里，连接问题变得明显了。Web Publishing Wizard 几乎总是不能找到文件正确的目标 UNC（统一命名约定）。原因很简单：大部分情况下该名字都被服务器隐藏起来了。你需要给自己的存贮目录提供一个完整的 UNC，如图 11.2 所示。确保提供的是一个 UNC 名，而不是标准的 DOS 驱动器和目录位置。使用 UNC 的原因是，它允许你使用同一种登记项技术，而不管服务器使用什么文件系统。

14. 单击 Next，Web Publishing Wizard 显示下一页，该页简单说明要验证你所提供的信息。单击 Next 开始验证过程。这次，你将会看到一条操作成功的消息。

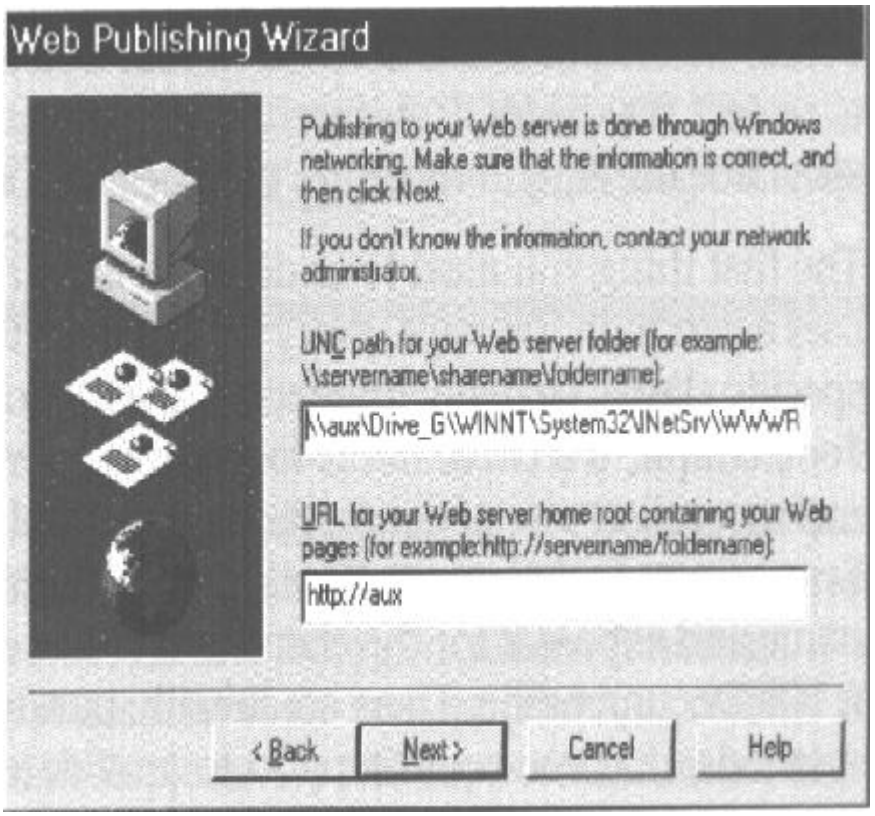
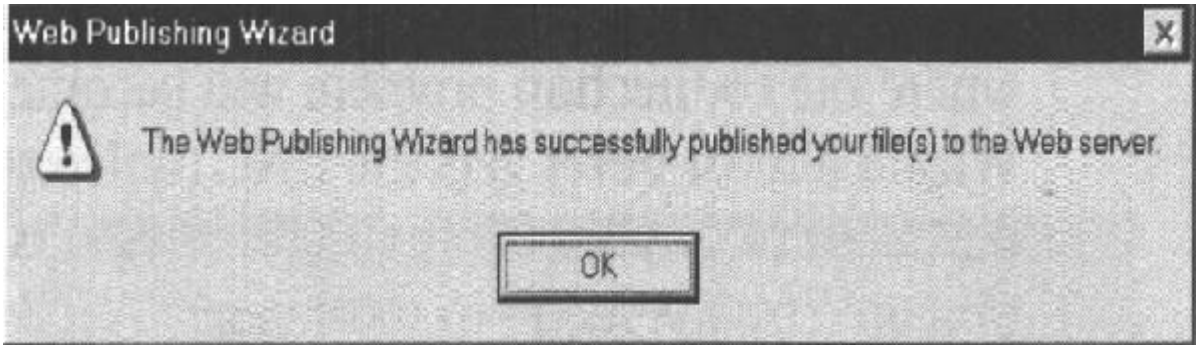


图 11.2　**本页显示了你配置** Web Publishing Wizard **时几乎所有问题的产生根源**

15. **单击** Finish**，完成文件传输过程。在** Web Publishing Wizard **为你拷贝完文件时，你可以看到一个文件传送对话框。文件传送完后，可以看到如下图所示的成功对话框。此时你已经修改了该文档，访问** Web **站点的任何人都能自动看到所做的修改。**



一旦你首次完成上述设置步骤，用户就可以只用简单的四步完成文件传送过程。而你需要做的就是将文档保存到本地驱动器上。利用 File|Send To|Web Publishing Wizard 命令启动该 Wizard，选中文件并选择一个连接，然后单击 Finish 就可以完成整个过程（即上面所列步骤 1 到步骤 4）。

技巧 网络管理员可以在每台机器上只执行这个任务一次，用以减少用户的支持电话。遗憾的是（至少到本书写作时），还没有办法像修改注册表或向一个目标机器拷贝文件那样自动完成上述任务。

11.2 ActiveDocument 结构概述

理解自己所使用控件的工作方式十分很重要的——至少要在一定程度上理解——只有这样在出现问题时你才能够解决它。本书无法详细介绍 OLE 的基本理论； 即使整本书都只讨论这个问题，恐怕写上 1000 页都还不够。不过，ActiveDocumcnt 控件只需要 OLE 的一部分特殊特性，这就是本节要介绍的内容。

首先需要理解的是，OLE 控件利用一个公用接口进行工作。你使用的每个 OLE 控件都必须支持特定的类（实际上是接口元素），以便让应用程序访问它。例如，如果客户机需要服务器提供在位操作支持，那么它就会调用与 IOleInplaceActivateObject 相关联的方法之一。每个支持在位操作的服务器都必须提供该名字的类，并提供给客户机使用。本例中，客户机的等价类是 IOleDocumentSite。实际上，你可以支持几百种类（及相关方法）以便提供各种 OLE 功能（不过不必支持所有类，因为某些形式的功能是专用控件不需要的）。图 11. 3 显示了在 Excel Worksheet 中使用的 OLE/COM Object Viewer（在第九章中已经介绍过这个实用程序）类（接口元素）。请注意，每个类都以 I 开头，而不是通常使用的 C。I 表示接口类，即客户机访问它们以获得服务器的某种功能。

注 由于 0LE 控件使用客户机了解和使用的标准接口，所以 0LE 控件能够工作。

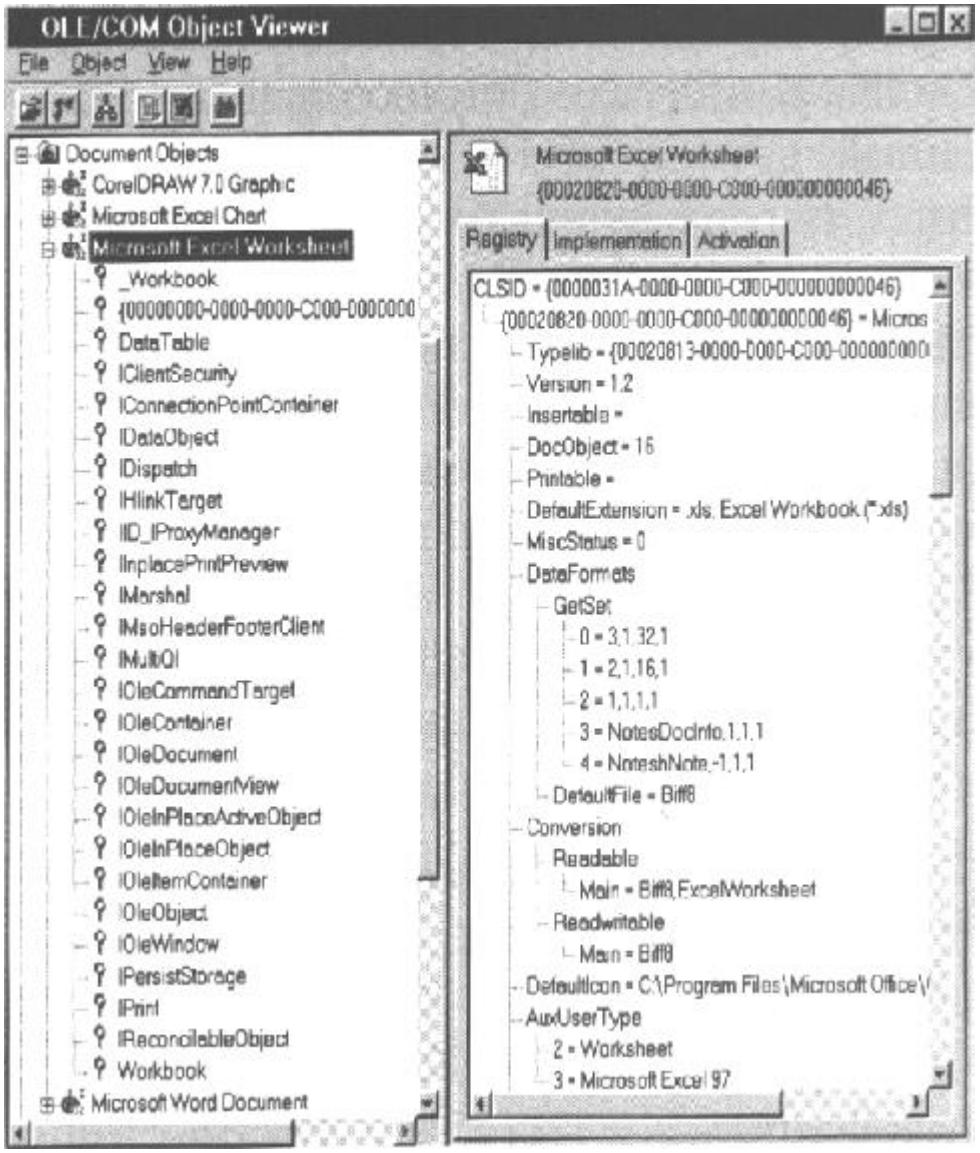


图 11.3　OLE/COM Object Viewer **提供了一种浏览创建控件时需要支持的接口元素方便的方法**

注释　OLE/COM Object Viewer 的老版本称为 OLE/COM Object View。这两个实用程序都能提供同样的信息。

注释 图 11.3 显示了 Microsoft Excel 的 ActiveDocument 的最新功能，你可以看到四个 IMso 函数，这是 ActiveDocument 函数的预定义名称。ActiveX 规范现在声称这些是标准的 IOle 类。幸运的是，好象 Internet Explorer3.0 将支持使用两种接口的文档对象。不过，这种支持不会持续很长时间的，你最好还是使用本章讨论的 IOle 版本调用。

显然，试图记住你是否在自己的控件中实现了所有这些调用是不现实的。幸运的是，像 Visual C++这样对建立 ActiveX 控件（或 OCX）提供直接支持的 IDE，通常负责创建大量的类。（现在已有很多编译器厂家为他们的产品创建 ActiveX 附加功能。例如 Borland 的 Delphi 就已经包含了一个 Internet 控件包附加功能，以帮助你建立适用于 Internet 的应用程序）。另外，他们还为每个所需类实现了一种缺省动作。只有当你需要加入某种编译器不支持的特殊功能时，才需要改变它的动作。在第 10 章中你已经看到，如果建立了一个合适的应用程序框架，那么建立一个简单的控件并不需要做太多工作。本示例在加入一行代码之前，花费那么大力气来设置好一切配置，其原因正在于此。

实际上，有一种简单的方式来讨论创建 ActiveDocument 所需的 OLE 接口函数。我们可以用本书中学到的知识来总结 ActiveDocument 的功能。如果希望自己的应用程序支持这种规范，就必须遵守下述四条原则。

- 实现 IPersistentStorage 。应用程序必须支持该类及其相关方法以便能使用 OLE 复合文件作为存贮介质。
- 支持 OLE 文档嵌入特性。该特性可以用多种方式实现。目前流行的方式是

通过菜单功能给用户提供两种嵌入方法：Insert Object（插入对象）和 Paste Special（粘贴特殊项）。用于完成这项工作的函数是 IPersistFile 、IOleObject 和 IDataObject。

- 提供在位(in-place)操作支持。你必须实现两个类来支持在位操作：IOleInplaceObject 、 IOlePlaceActivateObject 。为实现这些类，必须用 IOleInPlaceSite 和 IOleInPlaceFrame 类所提供的方法获取有关容器的信息。
- 加入 ActiveDocument 扩展。现在大部分 OLE 2 服务程序都能完成前面 3 步工作。为了使它能在 Internet 上工作，必须加入我们将简单讨论的这四个函数：IOleDocument、IOleDocumentView、IOleCommandTarget 和 Iprint。现在，让我们详细说明一下第四条。ActiveDocument 在某些方面相当新。由于 Microsoft 对这些规范很长时间没有公开，所以几乎很少有编译器能提供对 ActiveDocument 服务器的直接支持。也就是说，对于程序员，或者必须自己建立附加的接口（不很困难），或者升级编译器。下面四小节要介绍的内容是，为创建一个功能完整的 ActiveX 文档服务器，你必须要支持的四个附加的接口调用。（幸运的是，并不是所有这四个接口调用都需要——图 11.3 的显示说明,在 Excel 中,只需要实现其中三个)。另外 ActiveX SDK 也提供了创建这些接口元素所需的头文件和其它支持文件。本节也将对此作些介绍。

技巧 Microsoft Visual C++的4.2及以上版本，对于创建大部分ActiveX对象类型提供了直接支持，其中包括对ActiveDocument的直接支持。也就是说，这些版本实际上会创建我们本节所讨论的所有元素，从而减少创建自己新控件或应用程序时所需要的时间。

IOleDocument类

无论什么时候，当某个Internet客户看到实现了IOleDocument类的服务器时，它就会知道该服务器能作为一个ActiveDocument服务器。Internet Explorer和其它Internet客户机应用程序，在看到你的应用程序相关联的文档时，首先要查询的内容也正是这些。不要把标准的在位操作和ActiveDocument服务器所用的操作混淆起来。你可以建立一个只支持在位操作的应用程序，它在本地机器上会运行得很好，但它将不支持Internet上的在位活动。

注 IOle Document通知客户程序你的应用程序可以用作ActiveDocument服务器并提供了通讯服务。

注释 许多情况下，是否实现该接口并没什么关系，因为没有它，你的应用程序也能支持OLE2。忽略该接口的服务器，即使它对类似Word for Windows这样的本地客户支持在位操作时，也只是简单地在一个独立的窗口中被打开。

那么，除了让Internet Explorer知道应用程序支持ActiveDocument之外，该类还能做什么呢？每当客户需要创建新的服务器视图（CreateView方法）、枚

举视图（EnumViews 方法）、获取与 ActiveDocument 相关的 MiscStatus 位（GetDocMiscStatus 方法）时，都会调用这个类所支持的方法。实际上，这个类能帮助你从整体上管理服务器。它提供了客户机和服务器进行通讯的底层功能。

服务器视图并不等同于用户单击某个连接到你的文档链接时用户所看到的视图（那是由下节要介绍的 IOleDocumentView 界面管理的）。服务器视图提供的是服务器自身的一个单一实例。客户应用程序利用该视图进行通讯。例如，如果客户机需要找出服务器支持哪些特性，它就会利用服务器视图来完成。在介绍实际的应用程序代码时，你会很清楚地了解这一点。

注 服务器视图是服务器自身的一个单一实例——它支持在客户机和服务器间进行通信。

存在四种标准的混合(miscellaneous)状态位。下面逐一详细介绍。

- DOCMISC_CANCREATMULTIPLEVIEWS 让客户机知道服务器是否能创建多个视图。也就是说，该位定义了你是否能同时运行应用程序的多份拷贝。大多数情况下，现在的服务器都能做到这一点。唯一的例外是 CAD 或绘画程序，它们的内存需求实在太大了。另一个可能不支持该位的应用程序类是通讯程序，原因在于绝大多数人都只有一个调制解调器。
- DOCMISC_SUPPORTCOMPLEXRECTANGLES 用于指出服务器能否支持复杂的视图区域命令。例如，服务器是否允许客户机确定滚动条和缩放框的位置。
- DOCMISC_CANOPENEDIT 用于向客户机指出，服务器能否打开一个文

档用于编辑。设置该位能防止用户在线编辑文档（当你基于安全方面的考虑，或者希望创建一个只允许浏览的服务器时可以这么做）。

- DOCMISC_NOFILESUPPORT　用于指出服务器是否支持任意类型的文件操作。设置该位一般会使客户机显示一条错误消息，原因在于用户甚至不能读取所选的文件。

IOledocumentView类

与 IOleDocument 类相似，你必须实现 IOleDocumentView 类才能使用 ActiveDocument。这个特殊的接口元素依赖于 IOleDocument 类。打开文档前必须已有一个服务器在运行。另外，客户机通过从 IOleDocument 类的 GetDocMiscStatus 方法获得的信息，才能知道如何与该类进行交互。IOleDocumentView 类的每个拷贝都控制着一个 ActiveDocument 视图的实例。大部分情况下，这意味着 IOleDocumentView 类的单个实例控制着单个文档。不过，同样可以很容易为自己打开的文档的每个视图创建一个类的实例。

注　服务器视图是服务器自身的单一实例——它支持客户与服务器之间的通讯。

IOleDocumentView 类支持许多方法。表 11.1 列出了最普遍应用的方法。可以看出，这些方法允许你改变屏幕大小、创建你正在看到的视图的另一份拷贝，重置视图的边界区域或者确定哪个文档在当前视图中显示等。

表 11.1　与 IOleDocumentView 视图类项关联的方法

方法	动作
SetInplaceSite	将视图站点对象与相应视图联系起来。客户机提供视图站点对象。本质上，这是将文档与当前视图联系起来的方法
GetInplaceSite	返回指向与视图相关联的视图站点对象的指针
GetDocument	返回指向与视图相关的文档的指针
SetRect	定义视图的边界区域。也就是说，该方法设置用户将看到的窗口的大小
GetRect	返回视图的边界区域坐标
SetRectComplex	定义视图的复杂边界区域。该方法不仅确定用户将看到的窗口的大小，而且确定诸如滚动条和其它视图元素的位置。视图不一定非要支持该特色。如果你的应用程序不提供这种支持，就需要设置一个混合状态位（参见 IoleDocument 类描述中的详细介绍）
Show	让客户机显示或隐藏视图
UIActivate	确定用户接口是否活动。正常情况下，仅在视图得到焦点时用户接口才是活动的，其它情况下，用户借口均不处于活动状态，这样可以防止与已经得到焦点的视图冲突
Open	请求服务器在另一窗口中打开视图。可以用混合状态位关闭该特色（参见 IoleDocument 类中的详细说明）
CloseView	关闭视图

续表

SaveViewState	将当前视图状态信息写入 Istream
ApplyViewState	请求视图将其状态设置为先前保存在 IStream 中定义的设置值
Clone	创建当前视图的一个拷贝。复制的视图与原视图有相同的上下文关系，但使用不同的视图端口（IoleDocumentView 类的实例）

IOleCommandTarget 类

在生成 ActiveX 文档时，可以不实现 IOleCommandTarget 类。但它却不仅仅是为了方便才设置的类。IOleCommandTarget 类允许客户机和服务器进行交谈，而不必借助于像设置固定的菜单 ID 那样进行工作，但这一通讯受到限制。例如，通讯仍然只限于某些命令（下面将进一步介绍）。为完成接口工作的这一部分，必须分成两个步骤进行。

注　IOleCommandTarget 类允许服务器和客户机进行通讯而不必使用像固定菜单 ID 这样的技术。

这种客户机与服务器通讯的第一个步骤，是找出服务器支持哪些命令。客户机必须利用 Query（询问）方法来完成。表 11.2 列出了服务器支持的所有命令以及它们的标识符的完整列表。首先要注意的是，大部分命令都是标准的菜单项。

表 11.2　IOleCommandTarget 类支持的常用命令

命令	标识符
Edit Clear	OLECMDID_CLEARSELECTION
Edit Copy	OLECMDID_COPY
Edit Cut	OLECMDID_CUT
Edit Paste	OLECMDID_PASTE
Edit Paste Special	OLECMDID_PASTERSPECIAL
Edit Redo	OLECMDID_REDO
Edit Select All	OLECMDID_SELECTALL
Edit Undo	OLECMDID_UNDO
File New	OLECMDID_NEW
File Open	OLECMDID_OPEN
File Page Setup	OLECMDID_PAGESETUP
File Print	OLECMDID_PRINT
File Print Preview	OLECMDID_PRINTPREVIEW
File Properties	OLECMDID_PROPERTIES
File Save	OLECMDID_SAVE
File Save As	OLECMDID_SAVEAS
File Save Copy As	OLECMDID_SAVECOPYAS

续表

非标准命令。该标识符询问服务器是否能完成下面三个任务：返回缩放值、显示缩放对话框以及设置缩放值。如果服务器支持这三项任务，则该标识符一般与 View 菜单命令（或等价命令）相对应	OLECMDID_ZOOM
非标准命令。该标识符获取服务器支持的缩放范围。如果服务器支持它，则该标识符一般与 View Zoom 命令（或等价命令）相对应	OLECMDID_GETZOOMRANGE
Tools spelling	OLECMDID_SPELL

客户/服务器通讯的第二个步骤是，使用 Exec 方法，客户机给服务器发送一个或多个 OLECMD 结构。每个结构将只包含一条命令、该命令所需要的输入参数以及存放调用返回信息标志的位置。你不必提供什么输入参数，因此结构的该部分应为 NULL。标准的选项请参见表 11.3。表 11.4 描述了 Exec 调用返回的标志。

表 11.3　标准 Exec 方法的输入参数

标志	动作
OLECMDEXECOPT_PROMPTUSER	在执行命令前提示用户输入一些信息。例如，使用 File Open 命令时，你就会使用该选项
OLECMDEXECOPT_DONTPROMPUSER	不要求用户作任何输入。例如，当用户请求打印文档时，可以使用该选项
OLECMDEXECOPT_DODEFAULT	当你不清楚是否要提示用户时使用该选项。这种情况下可以让应用程序执行缺省动作。大部分情况下这意味着将提示用户进行输入
OLECMDEXECOPT_SHOWHLP	根本不执行命令；而是显示它的帮助屏。如果你的 ActiveDocument 能提供另外的帮助按钮，就可以使用该命令

表 11.4　标准 Exec 方法的返回值

标志	动作
OLECMDF_SUPPORTED	视图对象支持请求的命令
OLECMDF_ENABLED	该命令可以使用且视图对象已把它激活
OLECMDF_LATCHED	该命令使用了一个 On/Off 开关并且现在开关被置为 On
OLECMDF_NINCHED	视图对象不能确定使用 On/Off 开关的命令的状态。大多数情况下，意味着该命令使用的是三态配置并且正处于不确定状态。例如，如果用户为某安装程序选择了一些子项但没选其它项的话，一个三态复选框就可能返回该值。（该复选框在屏幕上显示为灰色。）

Iprint 类

IPrint 是另一个你能够实现的可选类。该类允许某个对象编程打印。IPrint 支持三种方法：打印(Print)、获取与打印相关的信息(GetPageInfo)以及设置打印作业的初始页号(SetInitialPageNum)。三种方法中，只有 Print 方法接收任意个数的标志作为输入。表 11.5 列出了这些标志以及它们的用法。

注　IPrint 的作用不言自明——允许对象支持可编程方式打印。

表 11.5　IPrint 的 Print 方法支持的标志

标志	动作
PRINTFLAG_MAYBOTHERUSER	告诉服务器，客户机允许用户进行交互操作。如果没设该标志，任何打印请求都必须自己完成。大部分情况下客户机都允许用户交互——唯一的例外可能涉及批处理打印作业或者打印操作处于后台的情况
PRINTFLAG_PROMPTUSER	提示用户在标准打印对话框（类似于 Windows 支持的打印框）中输入有关打印作业的信息。例如，如果指定该选项，则用户可以选择拷贝份数。使用该选项还必须同时指定 PRINTFLAG_PROMPTMAYBOTHER-USER 标志
PRINTFLAG_USERMAYCHANGE-DPRINTER	允许用户修改打印机设置。不过有些情况下可能不希望指定该选项——比如在网络环境中用户就不太容易访问到打印机。使用该选项必须同时指定 PRINTFLAG_PROMPTUSER 标志
PRINTFLAG_RECOMPOSETODEV-ICE	告诉目标打印机进行打印作业重组，例如，如果目标打印机支持比目前打印作业指定的分辨率更高，打印作业就应使用更高的分辨率
PRINTFLAG_DONTACTUALLYPR-INT	测试打印作业，但并不真正创建任何输出。该选项允许你在不浪费纸张的情况下测试用户接口的功能
PRINTFLAG_PRINTTOFILE	将打印输出送往文件而不是打印机

11.3 创建 ActiveDocument

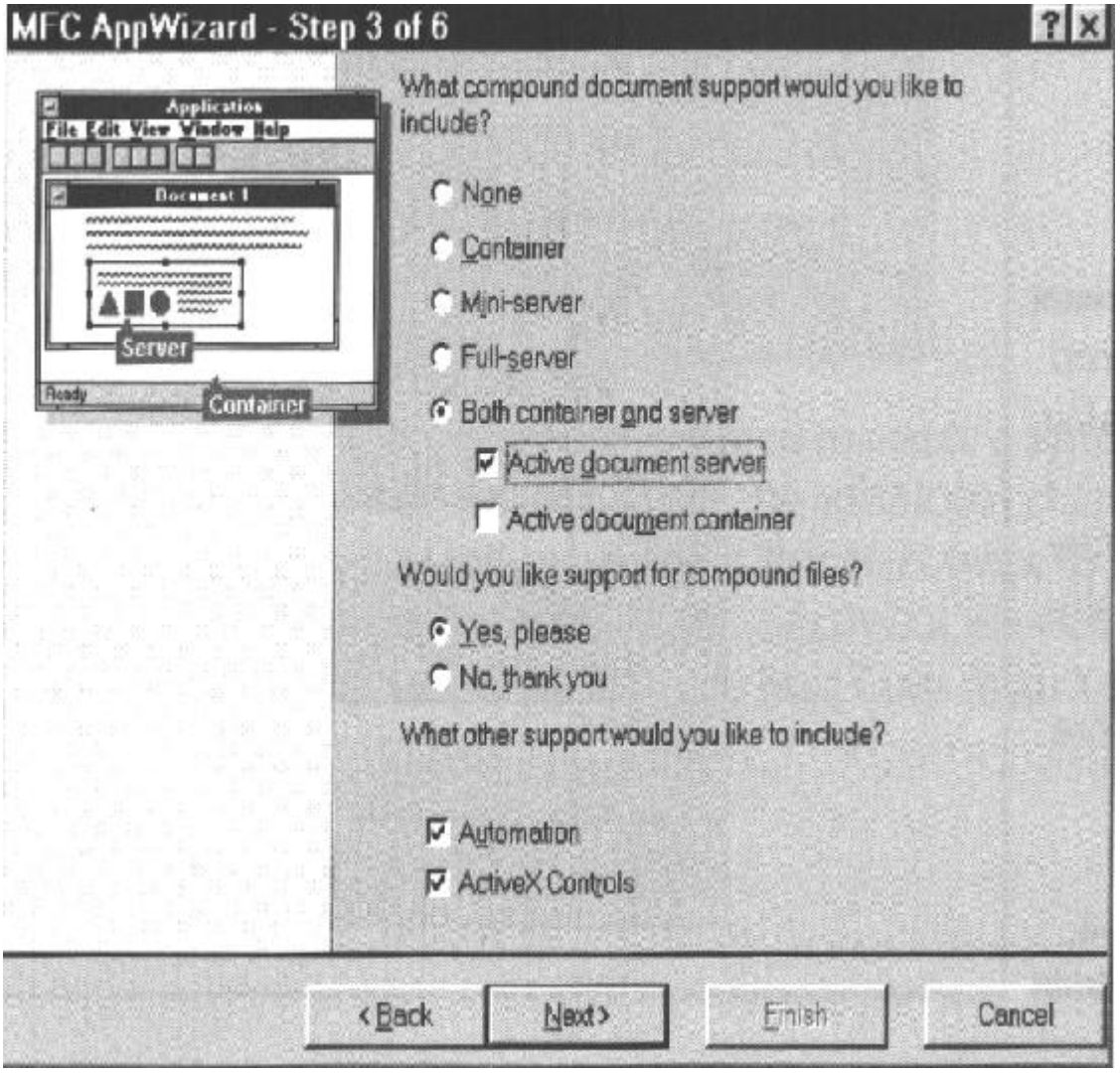
前面几节已经介绍了利用 ActiveDocument 应用程序（如 Word for Windows 或 Excel ）可以做些什么。如果你是一个只需要得到一些增强性能的高级用户，或者是一个只想优化自己的 Web 站点的 Web 管理员，那么，这些就足够了；但毕竟你是一个需要用这些功能创建应用程序的程序员。

即使对最好的程序员而言，手工加入前文所述的所有四个接口也是相当困难的。而且，由于 Microsoft 已经发布了 C++的新版本（4.2 以上版本都可以，不过 5.0 比 4. 2 更好用一些，6.0 版则已天衣无缝了），实际上也没有必要去找这个麻烦。本节要介绍的就是创建支持 ActiveDocument 的应用程序的快速方法。创建其它任何应用程序都有的步骤这里就不再详述了，而着重介绍工程中涉及 ActiveX 的特定部分。

首先，你需要完成的任务是创建一个新的工程工作区。第二章我们已经介绍了完成这项工作的步骤。你可以选择 MFC AppWizard，然后给应用程序命名——本例把它命名为 ActivDoc。单击 OK 开始一个创建过程，可以看到 MFC AppWizard 的第一页。为简单起见，选择 Wizard 第一页上的 Single Document 选项。两次单击 Next 跳过 Wizard 的第一和第二页，下面你将看到第三页对话框。

第三页是为应用程序完成大部分 ActiveDocument 设置的地方。你可以为应用程序提供五个不同层次的 OLE 支持，最后的三个层次允许你添加 ActiveDocument 支持。Mini-Server 选项不允许单独运行应用程序——必须在

Word for Windows、Internet Explorer 3.0 或其它什么容器中运行它。如果你想创建文件浏览器，这一级支持确实不错。下一选项是 Full-Server，它允许应用程序自己运行。你可以利用这种应用程序来支持对象，但不显示它们。绘图程序经常是服务器应用程序的好例子，但是，不必作为一个容器来运行。最后一级，Both Container and Server （容器与服务器）是本示例要选择的选项。它允许你在应用程序中提供完整的 OLE 2 功能，包括嵌入对象。另外，你还可以选择 ActiveX Document Server（文档服务器）复选框，如下图所示。另外，要确保选中 Automation(自动化）和 ActiveX Controls（控件）复选框。下图是选择选项后对话框的外观。



注释 ActiveX Document container（文档容器）是 Visual C++ 6.0 的新特色。它允许建立的应用程序包含其它的 ActiveDocument。例如，利用这种特色可以在没有真正提供 HTML 能力的应用程序中显示 Internet Explorer 文档。

单击 Next，**系统将显示** MFC AppWizard **的第四页。该页上的大部分设置都**

已完成。你可能想把 Recent File List（最新文件列表）的值设置的再高一点，因为大多数人都喜欢该特性（它当然要占用硬盘空间）。大部分情况下设为 9 或 10 就足够了；本示例中设置为 10。

该页还包括一个 Advanced（高级选项）按钮——如果不仔细的话，大部分程序员都会忽略该按钮。不幸的是，该按钮实在不应该标识为 Advanced，（也许 Microsoft 应考虑对 Application Wizard（应用程序向导）做点修改使某些设置的意义更明确一些）。单击 Advanced 按钮，即会显示如图 11.4 所示的对话框。

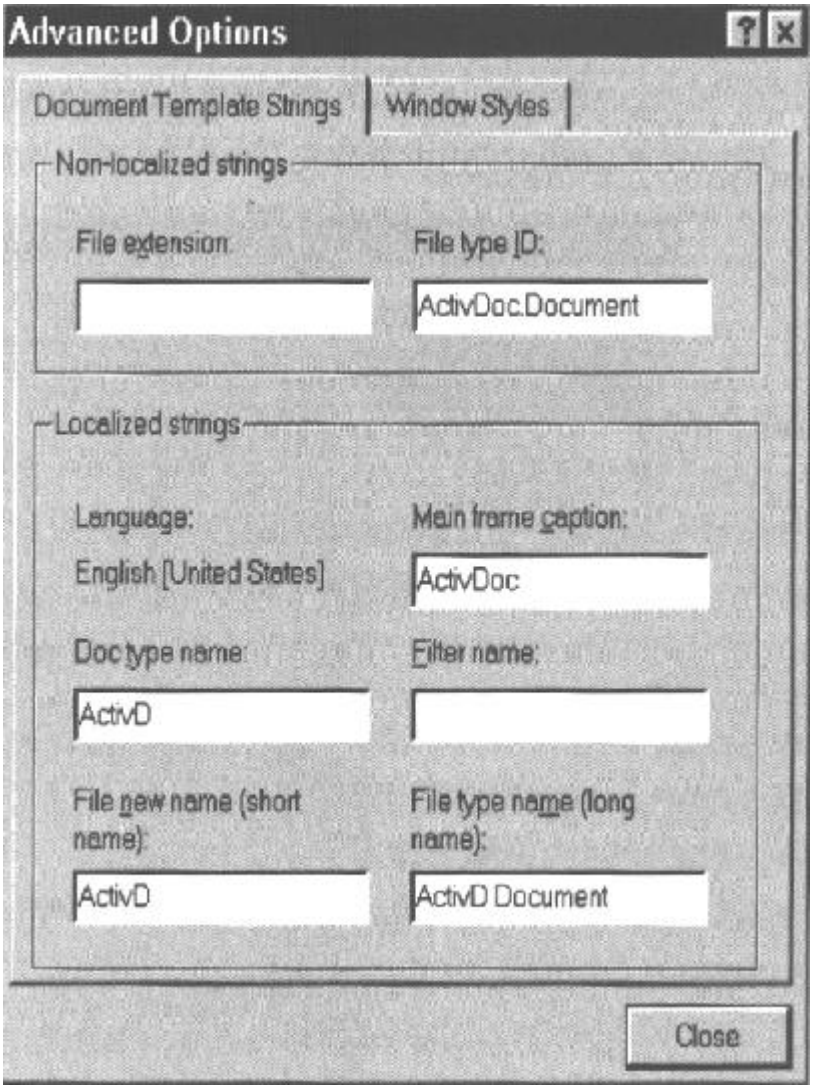
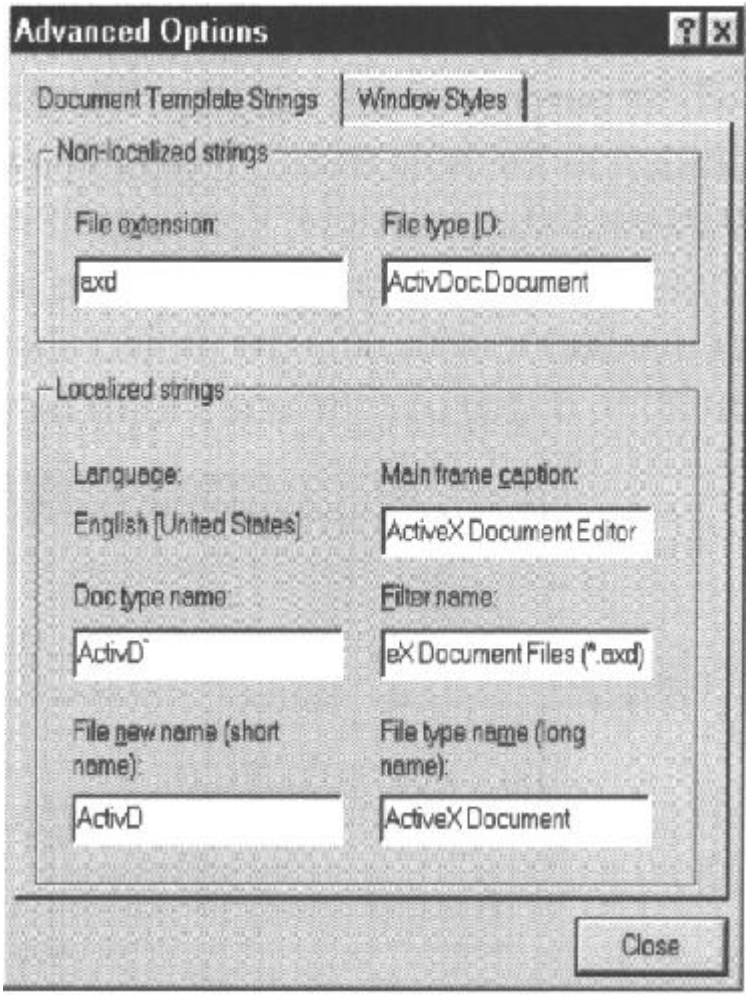


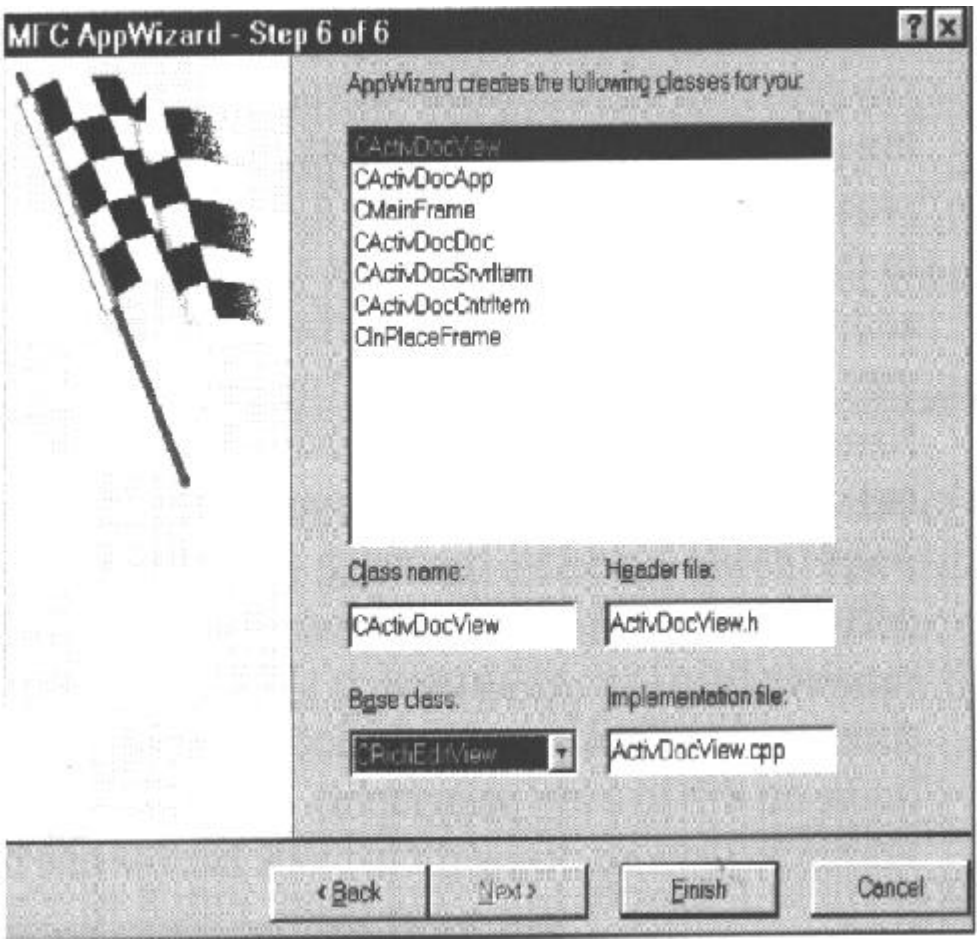
图 11.4　Advanced Options **对话框中至少包含了一个需要在创建应用程序前修改的选项**

Advanced Options**（高级选项）对话框的** Document Template Strings **（文档模板串）属性页允许你为应用程序设置文件扩展名。另外，它还自动完成一些幕后的工作。本例使用的文件扩展名为** AXD**。你要做的工作就是在第一个域（开始为空）中输入扩展名。也许你还想修改其它字符串，如** Frame Caption**（主框**

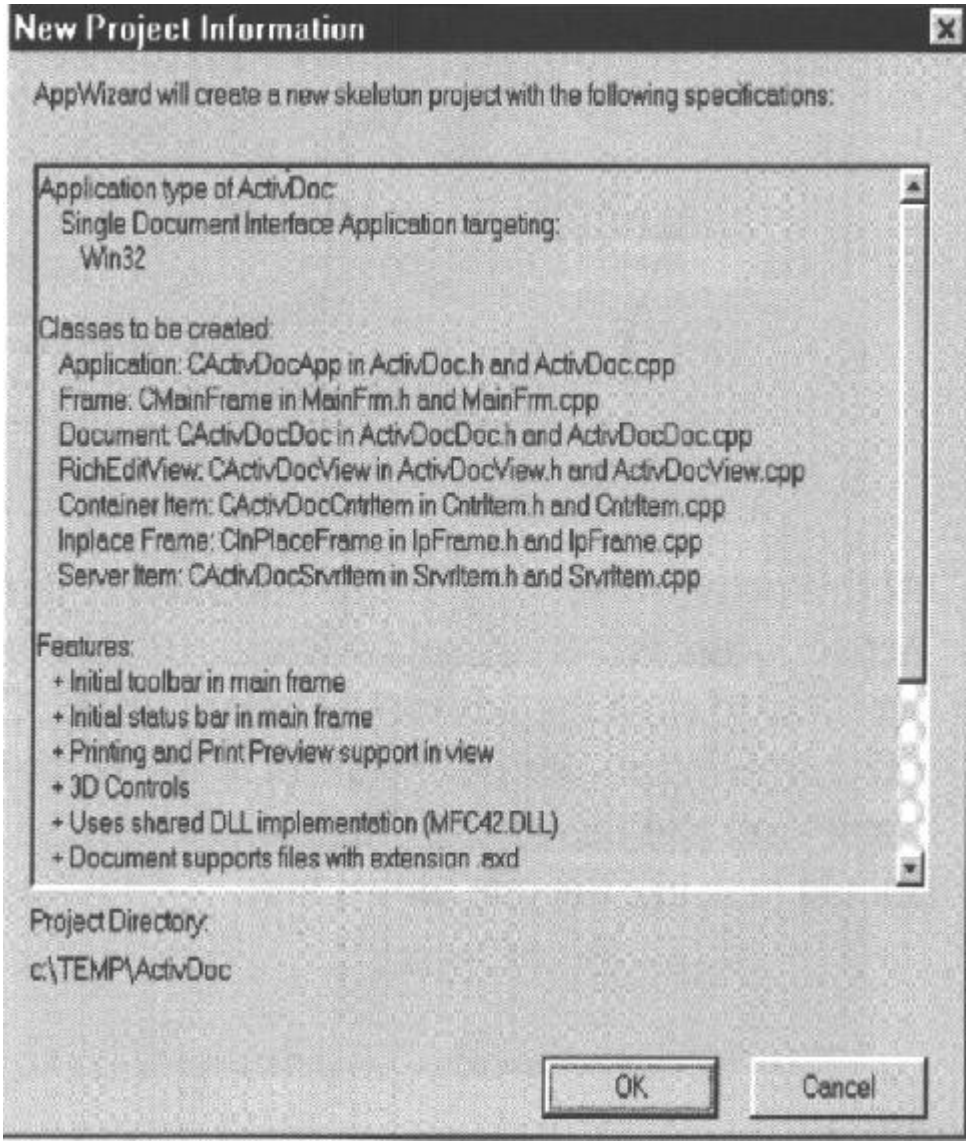
架标题）域。该例使用 ActiveX Document Editor（文档编辑器）。另外，你可能想让 Filter Name（过滤器名称）域中的输入值更富于描述性。初始时它是 ActivD Files(*.axd)，把它更改为 ActiveX Document Files(*.axd)时可读性会更好些。有些人确实不太在意 File Type Name（文件类型名称）域的长短（缺省为 ActiD Document），不过如果把它更改为 ActiveX Document 一定会对以后注册时的检索大有帮助。另外，它也是在 Windows 上下文相关菜单中用于显示你的新文档的字符串（稍后还将进一步介绍）。做完这些修改后，Advanced Options 对话框应如下图所示。



至此，我们已经完成了创建一个简单的 ActiveDocument **服务器的所有选项设置，但还有一点需要修改。单击** Close **关闭** Advanced Options **对话框。单击** Next **两次就可以到达** MFC AppWizard-Step6-6 **对话框。在类列表中选择** CActiveDocView **项，然后在** Base Class(**基类**）**域中选择** CRichEditView**。此时的对话框如下图所示。**



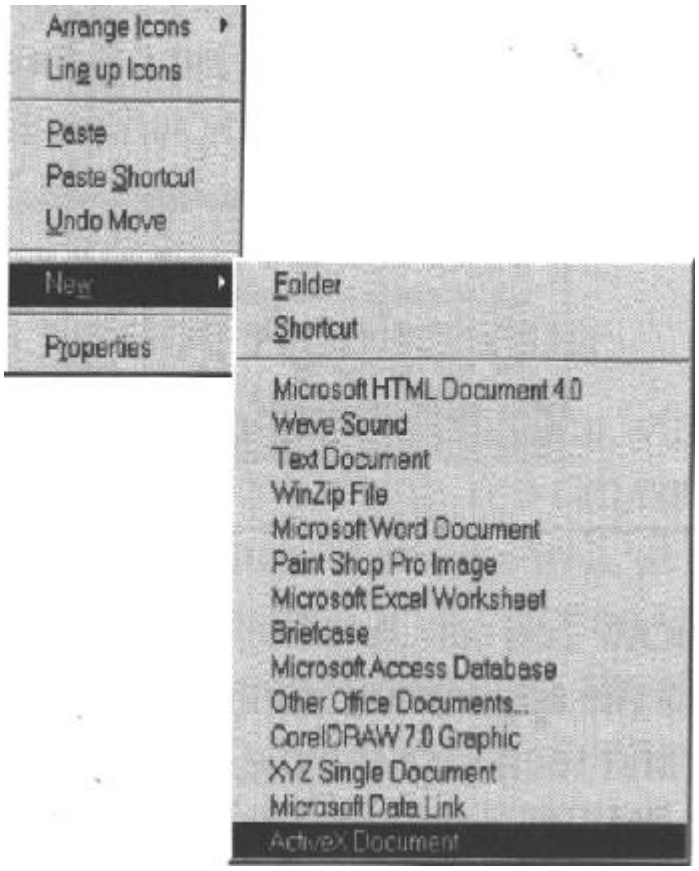
最后一步的目的是，在你愿意时，允许把该例子的应用程序作为简单的编辑器。如果愿意的话，你还可以实现标准的 CView **类，它不影响该程序作为** ActiveDocument **服务器的功能。现在单击** Finish **完成该工程，就会看到如下图所示的** New Project Information**（新工程信息）对话框：**



花点时间浏览一下特性列表，以确定应用程序所需的 ActiveX **支持是否完整（做过几个工程后，你就能通过该对话框迅速发现问题了）。单击** OK **生成工程。**

测试缺省的应用程序

现在，虽然我们的示例应用程序还做不了太多的事，但它确实能在 Wizard 之外做一些什么了。当 MFC AppWizard 创建完成之后，你可以编译并运行该应用程序。运行应用程序很重要，因为第一次运行时，应用程序将生成一些注册表项。可以发现的第一个变化是 Windows 上下文相关菜单现在包含了应用程序文件类型的一个表项，如下图所示，并且它与前面章节中我们在 Advanced Options 对话框的 File Type Name 域中输入的名称一样。



你还可以发现另一个变化。图 11.5 显示了在 OLE/COM Object Viewer 实用

程序（前面已用过它）中的 ActiveX 文档应用程序类型。可以看出，它与其它的文档对象——比如 Word for Windows——列在了一起。你应该能够立即发现，本章介绍的四个接口，在这里一个也没有出现，就像 Word for Windows 或其它 Binder 程序一样。在程序已经准备好进行测试之前检查接口，本节稍后会让你了解为什么这样做会大大节省时间和精力。

ActiveDoc 程序还能够创建一个基本的容器文件。你需要做的工作只是用 Insert/Object 命令给当前文档加入一个已有的对象。你可以将该文件保存到磁盘中。现在试着创建一个容器文件，以便能用 Internet Explorer 测试应用程序框架。确保已插入了一个对象，然后保存文件，否则打开文档时将什么也看不见。该例使用 ColorBlk. BMP 作为对象；而文件本身保存为 TESTDOC.AXD。一旦你创建了测试文档，就应该创建一个 HTML 页面来测试它。程序列表 11.2 显示了该例使用的代码。

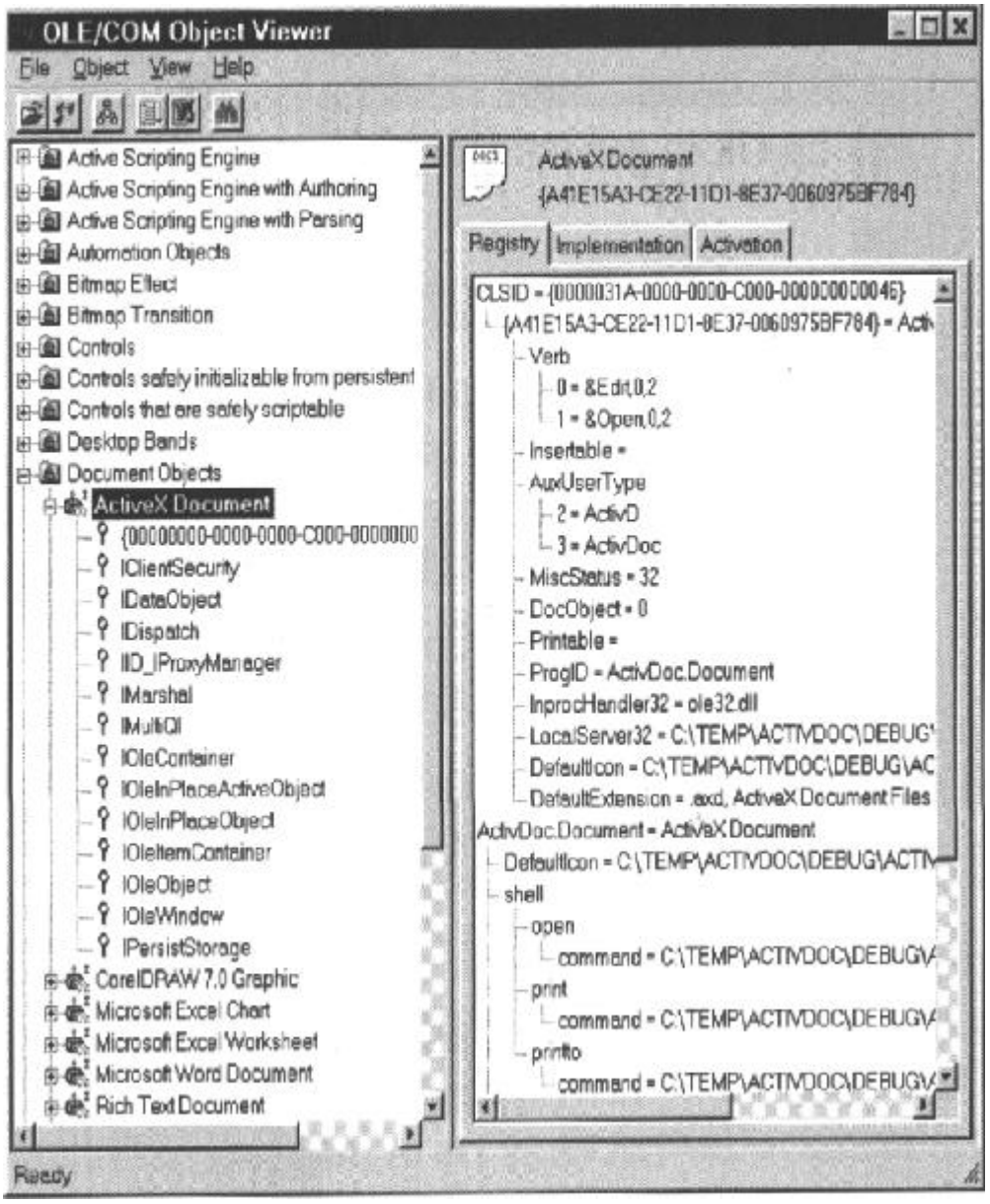


图 11.5　ActivDoc **应用程序作为** Document　Object　**（文档对象）显示在** OLE/COM Object Viewer **实用程序中**

程序列表 11.2

```
<HTML>
<HEAD>
<TITLE>new Page</TITLE>
</ HEAD>
<BODY>

<!-显示标题->
<CENTER><H2>ActiveX Document Test</H2></CENTER>

<!-创建一个到测试文档的链接->
Click < A HREF="TestDoc.AXD">Here</A> to test the AXD file

</BODY>
</HTML>
```

既然我们已经有了一个测试基础，那么让我们看一下应用程序的工作情况。在 Internet Explorer 中打开 Web 测试页，然后单击测试链接。你看到的是在 Internet Explorer 中我们的测试应用程序的一个副本，如图 11.6 所示，就如同在 Word for Windows 中工作时一样。请注意示例应用程序已经接管了浏览器的菜单和工具栏，就如同在 Word for Windows 文档中一样。另外，使用菜单可以插入新对象，还可以执行其它任务，就好象应用程序是在本地文档中工作一样。你还可以使

用 Web Publishing Wizard（本章前面已详细介绍过它）来保存这些修改。

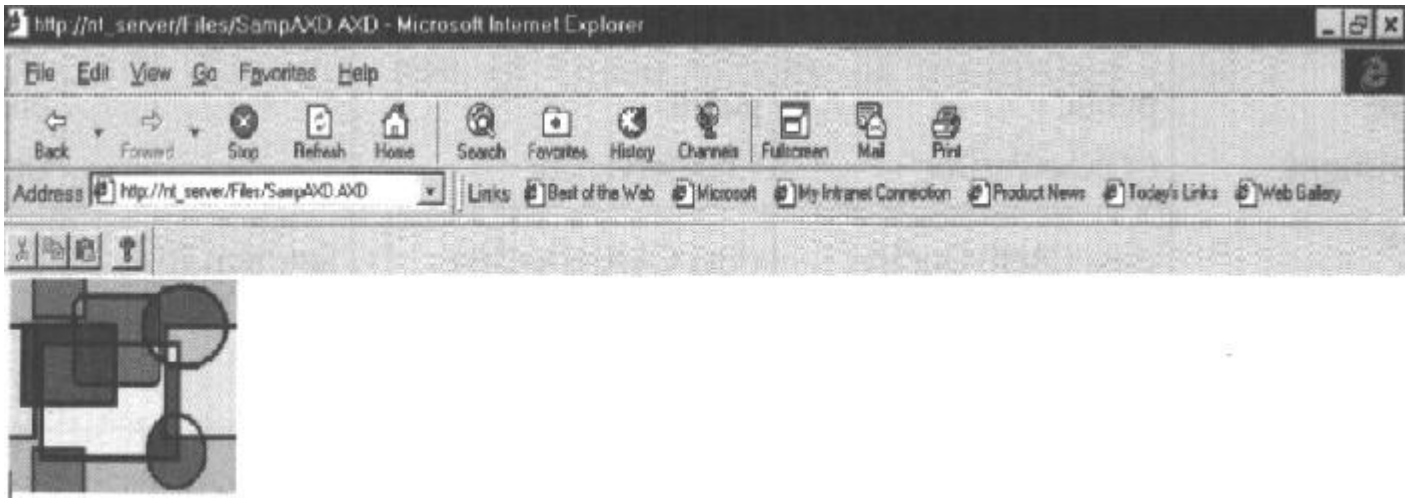


图 11.6　如任何 ActiveX 文档一样我们的文档显示在浏览器窗口中

注释　如果一切正常，系统会显示一个对话框，询问是保存还是打开文件（当然假定你在前面的会话中没有关掉对话框）。一定要告诉浏览器是打开文件，这样才能真正浏览它。

转换现存的应用程序

如果你有一个非常出色的应用程序，但它却不支持 ActiveDocument，请别担心，只要稍微做点工作就能将它转换成能够提供这种支持的应用程序了——至少比从头开始重写该程序要省事多了。下面就将介绍转换过程的 5 个步骤。如下所述，你可以利用它将任何一个已有的 OLE 服务器转换为基本的 ActiveDocument 服务器。不过要记住只能提供基本的支持。某些应用程序会和

这个例子一样工作正常，有些却不行。在开始加入支持项之前，应再花点时间去看看本章“ ActiveDocument 结构概述”一节并记下要点。做完附加的支持工作后，一定要用实际的 Internet 环境而不是本地驱动器来测试它们。这样做能保证测试所有接口，并且保证应用程序不会因为使用了标准的 OLE 接口，而没有使用你真正想测试的 ActiveDocument 接口来工作。

步骤 1：实现所需的类

首先并且最明显的一步是实现创建 ActiveX 文档服务器所需的类。如果你比较一下前面创建的例子和其它 C++ 4.2 版之前的应用程序，就会注意到声明类的方式有点不同。这是因为 Microsoft 已经从原来的 MFC 类中导出了许多子类，并且给它们加入了一些功能。表 11.6 列出了哪些类作了修改以及作了什么修改。第一列是在 C++ 4.2 版之前的应用程序中使用的类声明。第二列是现在的声明。第三列告诉你影响了哪些文件。第四列则是在以前 MFC 类文件中可能会发现的另外的类声明——它们是基于 Microsoft Office Binder 声明。

表 11.6 Active 文档类声明

最初的类说明	C++4.2 以上版本的新说明	ActivDoc 例子程序文件	MFC 类说明
class CInPlaceFrame: public COleIPFrameWnd	class CInPlaceFrame: public ColeFrameIPWnd	IPFrame.H	class CInPlaceFrame: public CDocObjectIPFrame Wnd

续表

class CActivDocDoc: public ColeServerDoc	class CActivDocDoc: public ColeServerDoc	ActivDocDoc.H	class CActivDocDoc: public DocObjectServerDoc:
class CactivDocSrvr: public ColeServerItem	class CActivDocServr: public CDocObjectServer	SrvrItem.H	class CActivDocSrvr: public CDocObjectServerItem

在头文件中替换了这些声明后，你还需要在相应的 CPP **文件中替换它们。利用** Microsoft Developer **的检索和替换功能会使工作变得容易一点。你不必担心会漏掉使用新类名的地方，在你做完头文件中的修改后，编译器会自动指出不一致的地方。所以必须先做头文件修改并且在往下做之前好好检查。（注意，如果你想用** ActiveX SDK **和老的** MFC **文件集来实现** ActiveDocument **服务器，那么必须修改三个头文件，而如果用** Visual C++ 4.2 **以上版本只需要修改两个头文件）。**

步骤 2：添加一个简单的声明

第二步是给应用程序的 STDAFX.H **文件中添加一个简单的声明：**

```
#include<afxdocob.h>
```

该头文件包含了使应用程序支持文档对象所需要的所有声明。实际上，略

微看看该文件就很有启发，因为它准确地声明了四个 ActiveDocument 接口元素是怎样实现的。

步骤 3：修改注册表项

第三步是修改你使用注册表项的方式。你正在创建的应用程序已不再仅仅是一个在位（In-place）服务器，所以需要在 CWinApp.CPP 文件（本例是 ActivDoc.CPP 文件）中修改注册表项，把 OAT_INPLACE_SERVER 改为 OAT_DOC_OBJECT_SERVER。代码如下所示：

```
m_server.UpdateRegistry(OAT_DOC_OBJECT_SERVER);
```

步骤 4：修改语法分析映射(Parse Map)

第四步是修改某些语法分析映射，因为必须让应用程序知道将打印命令和其它 OLE 相关信息送往何处。此时你需要对两个文件进行修改。第一个是应用程序文档头文件，本例中即为 ActivDocDoc.H。你必须加入下列代码行：

```
//Generated message map functions
protected:
    //{{AFX_MSG(CActivDocDoc)
        //NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //}}AFX_MSG
DECLARE_MESSAGE_MAP()
```

```
DECLARE_OLECMD_MAP()
};
```

如果消息映射包括 DECLARE_MESSAGE_MAP() **消息映射函数，就一定要在其后加入** DECLARE_OLECMD_MAP()**。第二个文件是应用程序文档** CPP **文件。在文件的一开头，你会看到至少有一个用于消息的映射区。还要加入另外一个映射区，如下所示：**

```
BEGIN_OLECME_MAP(CActiveDocDoc,COleServerDoc)
    ON_OLECMD_PAGESETUP()
    ON_OLECMD_PRINT()
END_OLECMD_MAP()
```

正如你所看到的那样，增加的代码允许应用程序使用 ID_FILE_PAGE_SETUP **和** ID_FILE_PRINT **标准标识符，在它们的处理函数中，通过安排打印函数来完成打印作业。你要做的工作只是给真正的句柄函数加入命令映射，如下所示：**

```
ON_COMMAND(ID_FILE_PRINT,OnFilePrint)
```

步骤 5：添加新函数

第五步（也是最后一步）用来给你的应用程序文档头文件及 CPP **文件增加新函数。**

一旦完成这些修改后，必须编译并测试修改后的应用程序。大多数情况下，

可以先在本地机上进行测试，以保证这些修改不会造成破坏。确信这些修改不会影响本地机性能后，可以试着从浏览器中打开文档。文档将在浏览器中打开而不是在另一窗口中（就像图 11.6 所示）。一定要保证该应用程序如同在本地机上使用时那样，能提供层次的功能（比如 File|Save 命令）。

第 12 章　使用 URL 和 Moniker

美国东北部的人们因为这样一句话而著称：“你不能从这儿直接到那儿，你必须首先途经另外一个地方。”有些人对 Internet 的看法也是如此，看起来总是要至少跨越四、五个 Web 页面，才能找到自己所需的起始站点。

正是这个原则，导致有些 Web 拥有者在他们的 Web 站点上，放置了许多链接。每个链接都指向与当前主题相关的信息——即从某个方面扩展该站点的信息。没有一个用户抱怨已有的这些链接，事实上，如果去掉这些链接他们反而会不满意。

不过当前这种链接系统确实存在一些问题。就像 Internet 上其它东西一样，它们都是静态的。不管如何进入站点，也不管是在什么样的条件设置下进行操作，对于一个特定的站点得到的链接集合总是相同。如果 Web 站点能有一点智能，从而可以帮助用户找到他们所需的东西，那时又会怎么样呢? 另外，如果用户能输入一些条件，使得从一个站点到另一站点更加容易，不是挺好吗? 例如，可以按照目前需求输入某种判断条件来定位某种特殊的站点。通过使用 URL（它定义出查找的目的地字符串）和 Moniker（它是由系统提供的实际的资源对象），ActiveX 提供了一种处理这种动态需求的方法。

注　统一资源定位器(URL)指出了在 Internet 上的什么地方进行检索。

注 在 Internet 上下文中，Moniker 是指用于保存位置对象的资源。

12.1 URL Moniker 概述

这一节中我们先介绍一下 URL 和 Moniker 这两个术语。URL（统一资源定位器）是一种方法，我们对它已经比较熟悉了，我们用它从一个站点走到另一站点。它就是使用浏览器时打入的 http://www.mycompany.com，URL 有三个主要部分，它们是协议（上例中是 http）、主机（www）和域（mycompany.com），你打入的 URL 的其余部分即是你标识的资源位置（如目录）。这种机制只所以非常有用，是因为可以无限制地扩展它。例如，如果你想用另一种协议，只需在 URL 的开头指明即可。Internet 有好几个这样的协议，包括 HTTP、FTP、Gopher 和 News。

注 URL 由三个主要部分组成：协议、主机名和域。

注释 本节中讨论的 URL，包括了全部的 URL，而不仅仅讨论一般情形。例如，一些 URL 把 host（主机）称为 home 或别的什么名称，如：http:/home.netscape.com。另外，一些域正变得越来越复杂了，从而可以提供更精确的信息，如，http:/www.wisconsin.edu.k6.us 是美国威斯康星州小学校的 URL。一些浏览器不要求用户打入协议名，他们简单地为资源核对所有合适的协议。总之，除了 URL 不同于他们提供的一般例子外，在这一节中，不对它作任何的假定。

Moniker 起源于 OLE 2 规范。实际上一个 Moniker 就是系统生成的一个对象，你可以利用该对象找到另一对象或者从中获取数据。某些定制的 Moniker 还支

持其它操作，但任何 Moniker 都提供两种基本操作。OLE 2 规范推出了同步 Moniker，也就是说应用程序必须等待，一直到系统已经从指定对象获取了所查找的数据为止。

Web 链接 如果你想在商业性的应用程序中使用 URL Moniker，最好先进行一点专业训练。Developer Solutions（开发者解决方案）在 http://www.devsolutions.com/ 上开设了一门课程，专门讲述 Moniker 的问题。三天的课程还讨论了各种其它与 OLE 相关的主题如 DCOM 和编组(marshalling)。如果你确实想了解Microsoft 提交给ANSI 申请批准的 OLE 标准的所有信息，可以浏览 http://info.gte.com/ftp/doc/activities/x3h7/by_model/ OLE 2.html。

将 URL 和 Moniker 的思想组合起来，就得到了 ActiveX API（从 Visual C++4.2 版至 6.0 版的一部分）提供的 URL Moniker。那么 ActiveX 使用的 URL Moniker 与 OLE 2 规范所说的同步有什么不同呢? 一方面，URL Moniker 同时提供了同步和异步捆绑，这在 Internet 上很重要，因为不知道要花多长时间才能找到所需数据。如果对 Internet 检索使用同步捆绑，应用程序将被挂起并给客户机造成许多问题（更不用说用户必须坐在那里等待，傻看着没有变化的应用程序）。URL Moniker 还提供了一种建立并使用 URL 的框架，这是 OLE 2 规范没涉及的领域。如果你要用标准的 OLE 2 应用程序访问远程数据，那么它必须支持网络上的 UNC（统一命名约定）设备定位。这意味着可能有某种映射的设备，但很显然，Internet 定位是不可能做到的。

创建 URL Moniker 需要三部分：客户机、系统和传输。客户机部分已经常驻于应用程序中了，本章“使用 URL Moniker”一节将进一步讨论客户机端。系统组件指的是操作系统的一部分（你不必担心这部分，因为 Microsoft 负责处理它）。传输简单地指出你想怎样将信息从服务器发送给客户机。这部分是可变的，它取决于你要实现什么传输方式。在 Internet 上一般使用 TCP。

注　URL Moniker 的三部分是客户机、系统和传输。

就像本书中到目前为止讨论的其它特殊的 OLE 功能一样，实现 URL Moniker 要求你创建或使用某些特殊接口。URL Moniker 的客户机、系统和传输三个组件中各有两个接口。下面将对每个接口一一说明。

注　和其它每种 ActiveX 编程技术一样，使用 URL Moniker 也需要增加特殊接口。

注　请谨记，Windows 还提供了实现 URL Moniker 所需要的两个系统级接口。

- IEnumFormatETC（客户机）　它是可选接口，允许你提供影响捆绑操作的特定协议信息。例如，要提供 MIME 功能，就需要提供该接口，以便列举应用程序所支持的格式。
- IBindStatusCallback（客户机）　传输将使用该接口来通知客户机的特定事件（如下载过程进度）。更重要的是，一旦从客户机经由系统到传输的捆绑完成，传输即利用 IBindSystemCallback::OnStartbinding 方法传回一个 IBinding 接口。
- IBindCtx（系统）　你在开始查找 Internet 资源时，都必须使用 CreatAsyncBindctx 这个 ActiveX API 调用，给 IBindStatusCallback 接口

（也可经给 IEmumFromatETC 接口）传送一个指针。它将返回一个指向 IBindCtx 接口的指针，负责完成客户机和传输之间的实际捆绑。

- IMoniker（系统） 该接口元素有许多用途。它的主要功能包括：使用 GetDisplayName 方法，按人们可读的格式，获取 moniker 名称，以及使用 BindToStorage 或 BindToObject 方法传输具体例子。
- IParseDisplayName(传输) 由ActiveX 支持的 MkParseDisplayNameEx() 函数的当前版本允许你根据文件或者 URL 创建 Moniker。不过你可能会发现，需要根据其它类型的对象创建 moniker。该接口允许你的应用程序与传输一起创建新的 moniker 类型。你还应该将这些新 moniker 类注册到注册表中。
- IBinding（传输） 系统创建一个由应用程序想要访问的协议类型所定义的传输。IBinding 就是产生的专用协议传输接口。它通过 IBindStatusCallback 接口，负责分析协议串、 驱动任何下载的数据、以及给应用程序提供状态信息。应用程序能利用 IBinding 接口启动、终止、暂停或恢复捆绑操作。

高级技巧

端口号与 URL 的关系

我不打算花费过多的时间去让你们学习那些可以留给工程师们解决的问题。但是，如果能在碰到硬件/软件连接问题时，知道一点关于 URL 中的“ http ”是如何工作的信息，那么，会使你工作起来更方便。Internet 上使用端口号的概念。也就是说，你在与 IP 地址 200.100.100.1 连接时，可以选择这个地址的某个特定端口。

实际上，这个概念是从人们在 Internet 上使用主机排斥一切的年代遗留下来的。你可以将某个终端插入主机背后的特定端口以获得特定服务。幸运的是，你可以对 PC 使用同样的概念。如同机器上的串口和并口一样，Internet 上的端口也有专用目的。

从软件角度也很好理解。任何在机器上做过汇编级编程的人都知道，必须访问特定端口号才能获取键盘信息。即使你没做过汇编程序，也可能使用特定端口来设置某个卡（如声卡）。任何想使用该声卡的软件（如游戏），都需要知道声卡的端口号才能发出声音。同样如果你想从 Internet 上获取特定信息，也需要端口号。

在 URL 中指定特定协议等同于申请特殊端口号。老的通讯程序要求你输入端口号；谢天谢地，URL 已经解决这个问题。例如，当你在 URL 中输入“http”时，实际上是请求访问主机上端口号 80。（其它如 TelNet 使用端口 23，某些交互游戏使用端口 4201。总之，每个协议都有一个特定端口号）。记住“http”比记住使用端口 80 要容易得多。由于 Internet 上的每个人都使用同样的端口号请求特定的公共服务，所以在 URL 中可以使用 http。

好了，现在来看看你为什么要知道这些信息。网络管理员并不总是使用服务的缺省端口号。你可以为 Internet 上特定的公司服务加入一个特殊端口号——只要服务器软件允许这样设置就行。尽管使用特殊端口号并不能代替防火墙和其它安全帮助，它仍不失为在 Internet 服务器上设定一块不容别人窥视的领地的一种好方法。在做某些配置工作如设置防火墙的过滤器时，知道端口号也会带来方便。最后，尤其对于程序员来说，如果一切看似正常，但却不能正常运行，那么了解 Internet 连接是如何工作的，将有助于解决应用程序的这种问题。

12.2 创建 URL Moniker

和其它类型的 OLE 通信一样，创建 URL Moniker 也遵循着一个相当标准的过程。这里存在着几级通信，可以选择是否完整地实现某些接口

(IEnumFormatETC)。下面的步骤使用最少的接口元素概述了一个典型的通信会话。换言之，该过程讲述的只是经常看到的东西，也包括了提高灵活性或提供附加信息交换等内容。

1. 创建捆绑环境。可以利用 ActiveX API 提供的 CreatAsyncBindCtx()函数。该调用要求为应用程序的 IBindStatusCallback 函数提供一个指针。没有该接口，系统和传输就没办法与应用程序进行通信。如果应用程序需要实现特殊功能，如 MIME，那么你还需要给可选的 IEnumFormatETC 接口提供一个指针。格式枚举器允许服务器和传输确定应用程序能处理什么数据格式（它们总假定应用程序能处理文本数据）。

2. 系统将一个指针返还给它的 IBindCtx 接口。该接口还通过捆绑环境将 IBindStatusCallback 进行注册。

3. 一旦应用程序有一个 IBindCtx 需要处理，它就能通过调用 CreatURLMoniker() 或 MkParseDisplayNameEx()函数来创建 URL Moniker。选择使用哪个函数很简单。如果你有 moniker，则使用 CreateURLMoniker()函数。否则就使用 MkParseDisplayNameEx() 函数来分析文本字符串，并根据它创建一个 Moniker，然后将信息传给 CreatURLMoniker()函数。

注释 Visual C++老版本的用户如果要实现 MkParseDisplayNameEx()函数，需要 MSDN 杂志提供的 Win32API 升级版，这种支持已经封装在 Visual C++ 5.0 和 6.0 软件包中了。

4. 下一步是利用系统的 IMoniker 接口创建能处理 URL Moniker 的传输。有两种方法可以使用：BindToObject()和 BindToStorage()调用允许你将指向

BindToObject() 调用允许你将指向 moniker 的对象实例化。该对象允许你与 Moniker 进行交互。一般是在 Web 站点上或其它活动连接中使用它。调用 BindToStorage()可获取由 moniker 指向的数据并保存它。该过程通常称为“捆绑”Moniker。实际上是你获取信息并存放到磁盘上的。你使用该调用类似于使用 FTP 站点。两情况下都要把在第二步中获得的指针作为调用的一部分传给 IBindCtx。

5. 系统的 IMoniker 接口将为你装载一个专用传输服务程序，而不管你请求什么样的协议。它会根据你传送的 Moniker 来确定使用什么协议。一旦专用传输服务程序被激活，它就使用 IBindStatusCallback::OnStartBinding()方法将指向 IBinding 接口的指针传送给应用程序。

6. 现在，在所有三个元素：应用程序、系统和传输间建立了通信，并且为应用程序提供了处理其它元素的两个接口（IBindCtx 和 IBinding）。应用程序能够开始下载文件、Web 页面或其它资源了。关于通信难题还有一点要说明。无论何时应用程序申请数据时，传输都会用 IBindStatusCallback::OnDataAvailable()方法告诉它下载的进度。应用程序利用该方法传递的状态信息确定下载是否完成（也可以用这些状态信息为用户更新进度栏或其它状态指示器）。

12.3 超链接基础

URL moniker 是从 Internet 上一个地方到达 Internet 上另一个地方的 OLE 实现的底层部分。还有另一个高层接口，它就是用户可以看到的接口。无论何时，如果在浏览器中看见带下划线的文本，那么就可能是看到了另一站点的超链接。

本书好几章都已经讨论了该过程的 HTML 部分（参见第八章 HTML 概述）。

注 URL moniker 接口的用户部分可以和 Web 页上的超链接一样简单。

Microsoft 提供了一种 ActiveX 超链接接口，你可以利用它模拟 ActiveX 控件内专用的浏览器活动。和其它形式的 OLE 一样，这意味着可以将浏览器技术扩展到应用程序中或者简单地用于 HTML 页面，以便给 ActiveX 控件提供另一附加级别的功能。无论在什么地方使用 ActiveX 超链接，都可以完成包括下述内容在内的一些基本服务。

- 应用程序到应用程序。没有浏览器时，你也能在两个应用程序间创建超链接。如果需要在自己的 LAN 上实现某种内部网站点，这种功能就特别方便。
- 应用程序到 ActiveX/HTML 文档。第 11 章已经介绍了 ActiveDocument 的工作原理。文档实际上是在浏览器窗口中显示的。使用标准的 HTML<A HREF>标记完成从 Internet 到文档的连接。如果能在应用程序中放一个控件，并且它能自动与 Internet 上的文档相连接，会是什么样的情况呢? 也就是说，该链接的作用等同于反向的 HTML< A HREF> 标记。ActiveX 超链接控件就能帮助你完成这个任务。例如，可以利用该功能提供从公司政策文档到 Internet 上其它信息源的链接。
- ActiveDocument 到 ActiveDocument。你一般是在浏览器中查看文档。如果能在文档中使用锚地，简单的从一个地方转移到另一地方（就像在 HTML 页上移动一样），事情会变得更好。ActiveX 超链接允许你完成这样的功能。你利用该功能从一个文档转移到另一文档，就像在 Internet

上从一个 HTML 页面转移到另一 HTML 页面一样。

- 从浏览器 ActiveDocument 到 Microsoft Office Binder ActiveDocument。公司在本地内部网上对使用该功能有潜在的需求。它允许用户把浏览器中显示的文档转移到本地机器上的一个 Microsoft Ofiice Binder 文档。例如，可以利用该功能提供与本地术语汇编的链接。无论何时，如果用户不知道某个词的意思，都可以沿着位于 Internet 上的链接，到达术语汇编的本地拷贝，从而弄清那个词的含义。

注　导航栈(navigation stack)是在当前会话中访问的 URL 列表，它由浏览器维护。

不过，在你使用上述任一种技巧之前，都必须首先创建 URL moniker。在前两节已经讨论过这种过程。另外，你还需要 ActiveX API 提供的超链接接口。幸运的是，你不必求助于诸如为 URL moniker 等提供的那些低级接口。如果你只是希望从 A 点到 B 点，你就可以利用 Microsoft 已经提供的下列四个函数来完成。

函数	描述
HlinkSimpleNavigateToString()	该函数允许你到达基于字符串的新地点，该字符串的格式与你在浏览器中输入的字符串格式一样。通常可用该函数到达一个新的 HTML 文档或者同一 HTML 文档中的另一位置，但它也适用于对象
HlinkSimpleNavigateToMoniker()	该函数利用 moniker 作为信息源，允许你从一个地方转移到另一地方，通常它适用于对象
HlinkGoBack()	该函数模拟浏览器工具栏上的 Go Back（退回）按钮。它将你送回导航栈所指的先前位置
HlinkGoForward()	该函数模拟浏览器工具栏上的 Go Forward（向前）按钮。它将你送到导航栈中的下一位置

技巧：这些简单的超链接函数都放在 URLHLINK.H 头文件中，（对 Visual C++ 5.0 以下版本的用户，这个文件由 ActiveX SDK 提供），你会在 URLMON.DLL 中看到实际的函数。

这些函数的使用是很简单的。首先介绍比较容易理解的函数：HLinkGoBack() 和 HLinkGoForward()，这两个函数都只需要一个参数，即指向当前文档的 IUnknown 接口的指针。可以利用 ExternalQueryInterface()函数来获取该指针，其中第一个参数是 IID_IUnknown，第二个参数是一个指针变量。

HLinkSimpleNavigateToString()和 HLinkSimpleNavigateToMoniker()函数稍复

杂一点。下面的命令行说明了它们所需的参数。

```
HLinkSimpleNavigateToString(szTarget, szLocation, szTargetFrame, pIUnknown,
    pIBindCtx, pIBindStatusCallback,grfHLNF, dwReserved)
HLinkSimpleNavigateToMoniker(pmkTarget, szLocation, szTargetFrame, pIUnknown,
    pIBindCtx, pIBindStatusCallback,grfHLNF, dwReserved)
```

除第一个参数外，这两个函数使用了同样的参数表。HLinkSimpleNavigateToString()的第一个参数是一个包含着将要访问站点的 URL（或其它标识信息）的串。HLinkSimpleNavigateToMoniker()的第一个参数是指向 moniker 的指针。

现在看看公共参数。szLocation 包含同一页上某个位置的名称。第 8 章已经介绍了怎样做到这一点。szTargetFrame 参数包含文档内某个框架的名称。它是 HTML 专用的，并且只适用于使用框架的页。这两个参数都不是必须的，仅在需要修饰一些扩展的导航信息时才需要提供它们。不想包括它们时可以将它们的值设置为 NULL。

另外，还有三个接口专用参数：pIUnknown，pIBindCtx，IBindStatusCallback。这里只有 pIUnknown 参数是必须的，它可以利用 ExternalQueryInterface()函数得到。如果你想提供另外的处理信息，就要指定 pIBindCtx 值。例如，如果你的应用程序提供 MIME 功能，就可能利用该参数传递 IEnumFormatETC 接口的位置（它是作为 IBindCtx 参数的一部分传递的，但是，不要给 IEnumFormatETC 接口本身传递指针）。如果在传送中需要状态信息，那么，还要给应用程序的 IBindStatusCallback 接口传递一个指针。

剩下的就是 grfHLNF 参数了。它允许你指定浏览器的特定行为。例如，如果将该参数设置为 HLNG_OPEN_INNEWWINDOW，那么浏览器会在新窗口中显示指定站点。在 HLINK.H 头文件（对于 Visual C++ 5.0 以下版本的用户，该文件由 ActiveX SDK 提供）中有 HLNF 值的完整列表。

12.4 理解超链接接口

在某些情况下，前面讨论的简单的超链接函数调用还显得不够用。例如，你可能想实现一个完整的浏览器，把它作为应用程序的一部分。另一种情形是，如果你想提供这些调用不支持的某些扩展功能，比如操作历史列表等，这时简单调用就显得不够用了。前面讨论的调用特别适用于只关心导航的情形。

对任何 ActiveX 控件，都需要提供一些附加的接口，以便在应用程序中实现超链接。实际上有五个新接口，如下所述：

- IHLink（必选项） 这是 ActiveX 超链接接口的中心。它是一个提供了另一个需要进行超链接的应用程序所需的所有信息的接口，包括目标 moniker（实际对象标识）、URL 风格的字符串以及一个友好的名称。它还能提供其它类型信息，但创建简单的超链接一般不需要。所有简单链接都是通过 Navigate 方法完成的，它正是前面讨论的简单函数使用的方法。
- IHLinkTarget（可选项） 利用该接口可以在文档中定位或者下载其它信息。例如，该接口允许你在电子表格中指向特定单元格。IHLinkTarget

还提供了一种将指向文档的 IBrowseContext 接口的指针传递给调用的应用程序的手段。应用程序可以链接到一个不提供支持该接口的某一文档，但也只能如此。

- **IHLinkFrame（可选项）** 记住，在 OLE 中，框架是 OLE 容器中的一个应用程序容器。它包含菜单和外部框。该接口允许框架维护与 OLE 容器（通常为某种文档）的联系。
- **IHLinkSite（可选项）** 这是管理超链接站点的接口。它有两项基本任务，并能提供并发服务。该接口的第一个主要任务是，获得对文档的 IHLink 对象的访问权。访问该对象能提供许多信息，如当前站点名。接口的第二个主要目的是，提供对客户机应用程序的反馈。例如，OnNavigationComplete 方法能告诉客户机下载或其它任务已经结束。
- **IBrowseContext（可选项）** 应用程序通过该接口获得对浏览器环境对象的访问。所有浏览环境对象的全部工作就是追踪导航栈——即当前会话期间访问的站点。前文所述的 IHLinkGoFoward()和 IHLinkGoBack()函数正是根据该接口来完成它们的工作的。

实际上，在这一组接口中有两级实现。我们认为第一级是应用程序，在这里是指浏览器。第二级是一个 OLE 容器，在这里是指文档。浏览器实现 IHLinkFrame 和 IBrowseContext 接口。这两个接口只会有一种情况发生，即浏览器要么实现它们、要么不实现。文档负责实现 IHLinkTarget、IHLink 和 IHLinkSite 接口。

用接口术语来定义文档有点困难。文档的容器部分实现 IHLinkTarget 接口，

所以，即使是处理一个包含多个对象的文档，你也只能看到其中之一。同样，无论文档包含多少对象，每个文档也只有一个 HLinkSite 接口。但你可以在一个文档中包含多个超链接。

让我们快速浏览一下超链接。每个超链接都需要单独的 IHLink 接口，可以用四个函数来建立：IHLinkCreatFromMoniker()、IHLinkCreatFromString()、IHLinkCreatFromData()以及 IHLinkQueryCreatFromData()。HLink 对象可以使用持久数据，也就是说控件必须实现一个 IPersistStream 接口。因为你可以从数据中创建一个 HLink，所以就可能利用 Clipboard 来剪切，拷贝并粘贴它。另外，因为 IHLink 可以使用 moniker，所以你可以指向任意类型 OLE 文档，包括那些一般不在浏览器中显示的文档，如 Word for Windows 文件。

12.5 使用 URL Moniker

今天，许多公司特别关注 Internet。因此，在不久的将来，访问 Internet 上的信息，一定会像访问本地硬盘数据一样容易。实际上，随着 Internet Explorer 4.0 中包含了部分 Visual C++ 6.0 特性，Microsoft 已经向上述目标迈进了一步。你将会发现硬盘和 Internet 上你最喜欢的站点正共享着一个公用的 Explorer 视图。

WEB 链接 获取自己需要的信息经常是指访问自己的本地新闻组，以获得其它的程序员的帮助。不必多说，你会发现有些新闻组比别人花更多的时间在讨论 URL moniker ；毕竟它还算是一个神秘的课题。例如，microsoft.public.win32

.programmer.ole 新闻组很长时间以来一直在讨论将 Moniker 捆绑到专用控件上的话题。甚至还有一个新闻组专门用于 URL moniker：microsoft.public.activex.programming.urlmonikers。如果你想从新闻组中的人那里获得Visual C++有关的帮助,可以看一看microsoft public.vc.mfcole。

当然，数据站点的这种界线模糊倾向(blurring)对用户和程序员都会提供一些新机遇。例如，作为用户你会发现，将你需要的数据放入你需要的应用程序中去进行编辑要省事一点。作为程序员你会发现，现在还有许多别的方法来处理一个程序，并使它产生特殊的效果。

注释 如果你不是使用 Visual C++5.0 以上版本来创建本节的例子，那么，就必须安装 Activex SDK。SDK 提供了能使示例代码正常工作所需的 URLMON.H MLINK.LIB 和 URLMON.LIB 文件。

使应用程序具有特殊效果的一种最普通方法就是给它添加超链接功能，而最容易的添加方法是，在应用程序的工具栏上，放置一个可以使用户进入公司内部网的按钮。同样，你可以使用另一按钮，允许用户选择特定站点或在公司授权的 Internet 站点列表中进行选择。这正是本节要介绍的内容。样本程序将说明如何给某个典型的应用程序加入两个按钮，从而使用户可以快速简单地链接到 Internet 上。

第一步当然是创建一个新的应用程序。就像第 11 章中的 ActiveDocument 例子一样，你也可以使用老版本的 Microsoft Visual C++编译器来创建这些代码。本节假定你使用的是 5.0 以上版本，尽管用 4.2 版本创建它相对容易些。由于

第 2 章已经介绍了使用 MFCApplication Wizard 创建应用程序的过程，所以这里只进行简短的描述。在创建过程中，你需要选择一些选项才能使例子正常工作。首先，例子的名称为 ViewURL（尽管你可以选择自己喜欢的名字）。其次，在 Wizard 的第 1 页应选择 Single Document 。第三，在 Wizard 的第 4 页上应选择 Windows Sockets 支持。最后，在 MFC AppWizard 的第 6 页将 Base Class 域设置为 CEditView，这将允许你在示例应用程序的主窗口中编辑文本。

技巧 还有许多情况像本例一样不需要 CRichEditView 类的全部功能。与 CEditView 相比，CRichEditView 类不仅相当大，而且使用 CRichEditView 类还要求给应用程序添加 OLE 容器支持功能，这就进一步增大了应用程序的规模。CEditView 类既能让你显示无格式文本，又能使应用程序比较小——这正是实用类应用程序一个很重要的考虑因素。

添加库支持

一旦 Wizard 完成了应用程序创建工作后，你就可以开始做进一步的修改了。首先要完成的工作就是给应用程序加入超链接支持。为此，可在 STDAFX.H 文件中加入下述的#include。

```
//Added for URL support.
#include "URLMon.h"
```

该头文件包含了你在使用各种 URL moniker 有关命令时所需的所有#define

语句。本章前面已经介绍了其中一些命令，这里将介绍它们是如何工作的。

注释 老版本的 Visual C++ 用户要做的下一件事就是添加一些静态的支持（但是 Visual C++ 5.0 以上的用户不必做这一步）。HLINK.LIB 和 URLMON. LIB 这两个库都在 ActiveX SDK 的 LIB 文件夹中。可以利用 Visual C++ 的 Insert （插入）| Files Into Project（文件到项目）命令给应用程序添加所需的库支持。通过查看 FileView 中的包括文件列表，来检查是否真的添加了支持（只需单击显示工作区左边的 FileView 标志即可）。

技巧 Microsoft 打算将来把 HLINK.LIB 和 URLMON.LIB 中的静态库支持放入 URLMON.DLL 中。一定要在编写支持 URL moniker 的应用程序之前检查该动态库的支持功能。

创建所需资源

现在已经加入了所需的库支持，让我们给工具栏添加一些按钮。单击 Resource View （资源视图），打开 Toolbar 文件夹，然后双击 IDR_MAINFRAME 项，就能看到如图 12.1 所示的标准工具栏。添加新按钮很容易，你只需单击工具栏末尾的空白按钮，然后开始在绘画区域中显示的空白按钮上绘制。把这个按钮稍微向右移动一点，就能将它与工具栏上已显示的其它按钮区分开。图 12.1 显示的是这个例子中添加的两个按钮。

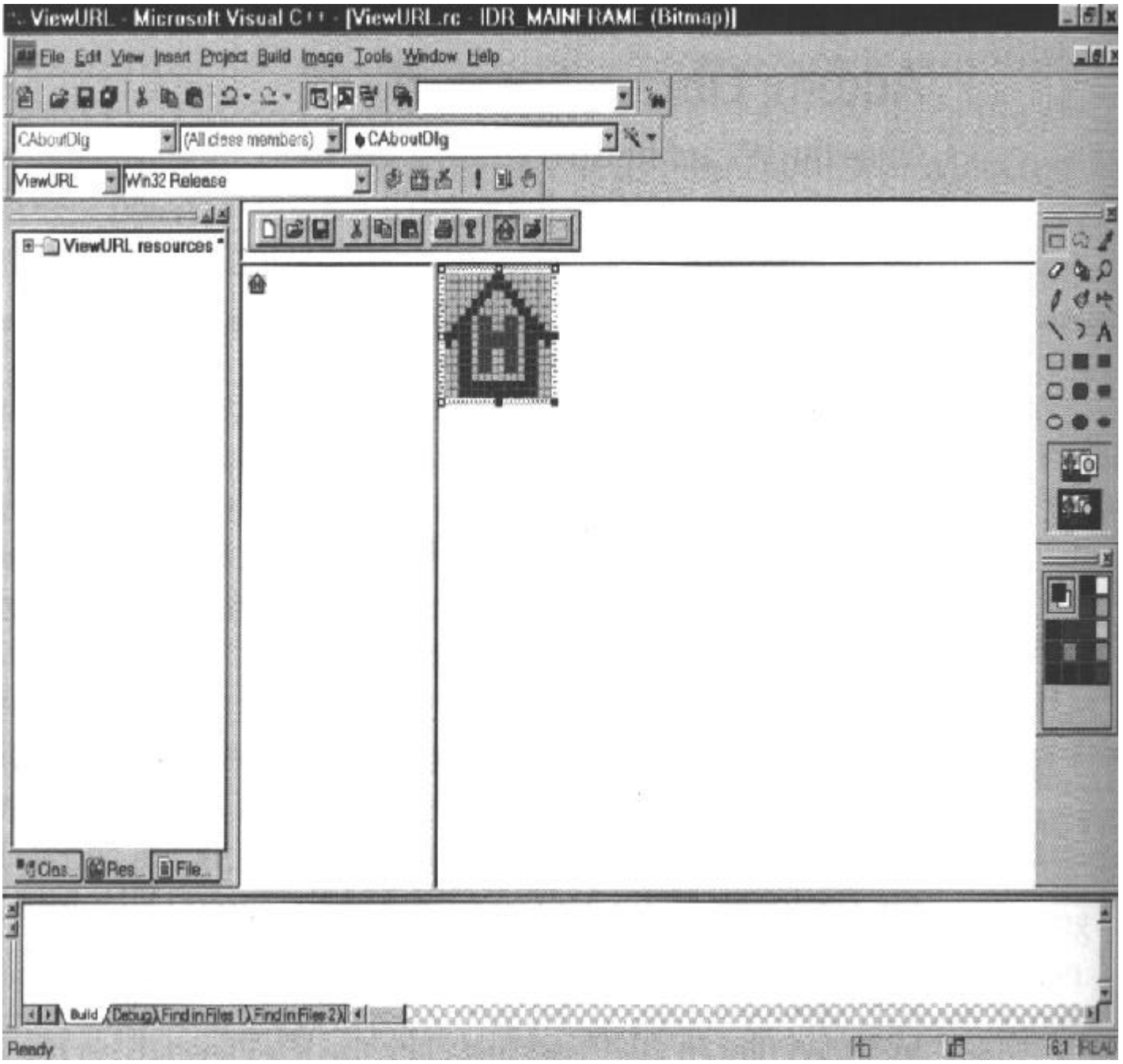
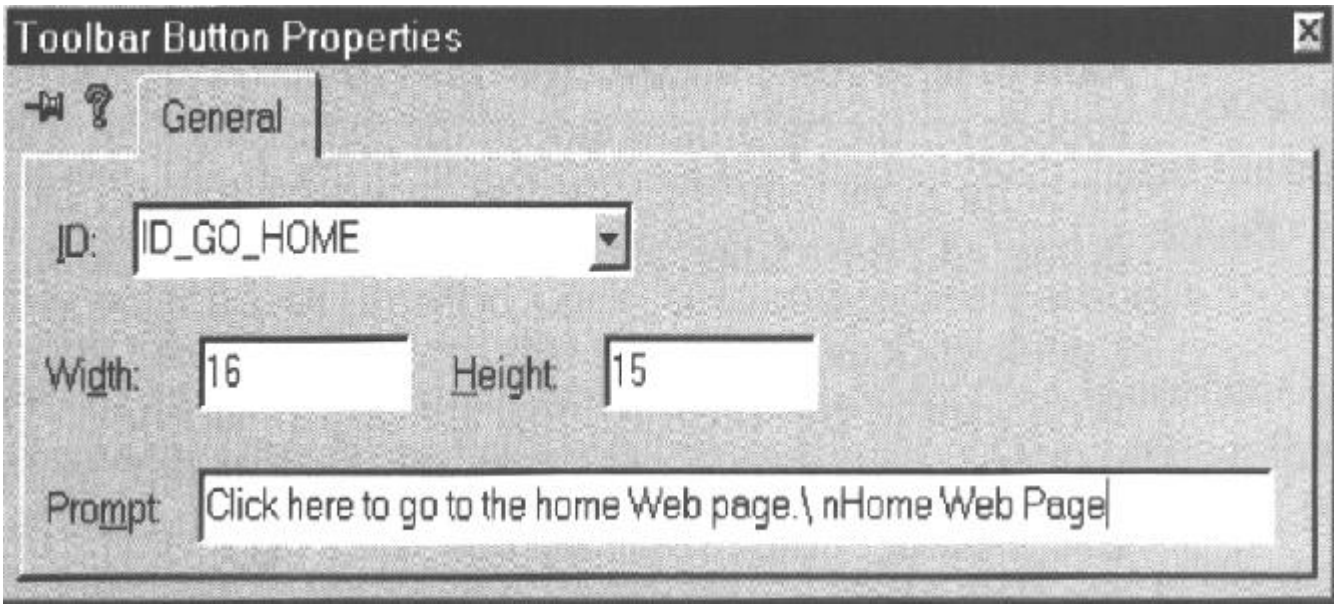


图 12.1　**具有** Home Page **和** Any Web Page **按钮的标准工具栏**

注释 增加按钮可以给用户提供一种可视图化的显示，另外，还要给应用程序添加一些按钮标识信息。简单地双击将要设置的按钮的工具栏（不要单击刚创建的按钮图标），就会看到如下图所示的对话框。对两个按钮使用下表进行的设置。

标识符	描述	高度	宽度
ID_GO_HOME	Click here to go to home Web page.\nHome Web Page	15	16
ID_GO_SITE	Click here to go to any Web page.\nAny Web Page	15	16

技巧 注意 Caption 包含了一个较长的描述、一个\n 以及一个较短的描述。较长的描述将出现在应用程序的状态栏上。较短的描述用于当鼠标停留在该控件时的弹出帮助。一定要用“ \n ”分隔这两项。在 Visual C++ 应用程序中任何看到弹出式帮助的地方都可以使用这种特殊的输入方法。



单击第一个按钮将把用户带入公司的主页。按钮已经起作用，在资源方面不必再做什么了。你需要添加的只是完成实际工作的一点代码。但是，第二个按钮将允许用户输入他们自己的站点，如果愿意的话，还可以输入该站点中的某个位置。该按钮还支持那些使用框架的站点上的框架，这就需要添加一个特殊的对话框。打开 ResourceView **中的** Dialog **文件夹，用右键单击** Dialog **文件夹，就能看到一个关联菜单。 选择** Insert Dialog **项，就能看到一个名为** IDD_DIALOG1 **的新对话框添加到了文件夹的列表中（应用程序当前提供的唯一一个对话框是** About Box **对话框）。**

现在给对话框改名。用右键单击 IDD_DIALOG1 **项，然后从关联菜单中选择** Properties**。在** ID **字段中输入** IDD_SITE_SELECT**，然后单击属性对话框，就能看到新名字出现在** Dialog **文件夹的对话框中。**

给对话框添加所需控件相当容易。首先要做的是将对话框大小改为 25×120

像素点（即出现在状态栏右边的当前对话框大小）。这可能要占一点空间，因为用户可能想输入很长的 URL。改变对话框大小后，可以添加三个标签和三个编辑控件，如图 12.2 所示。

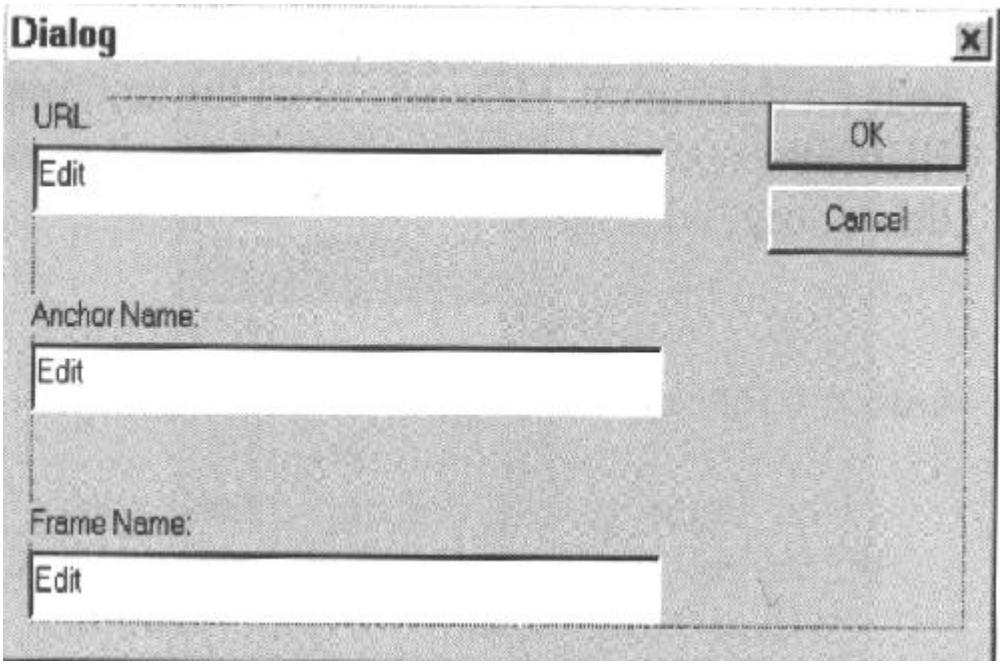
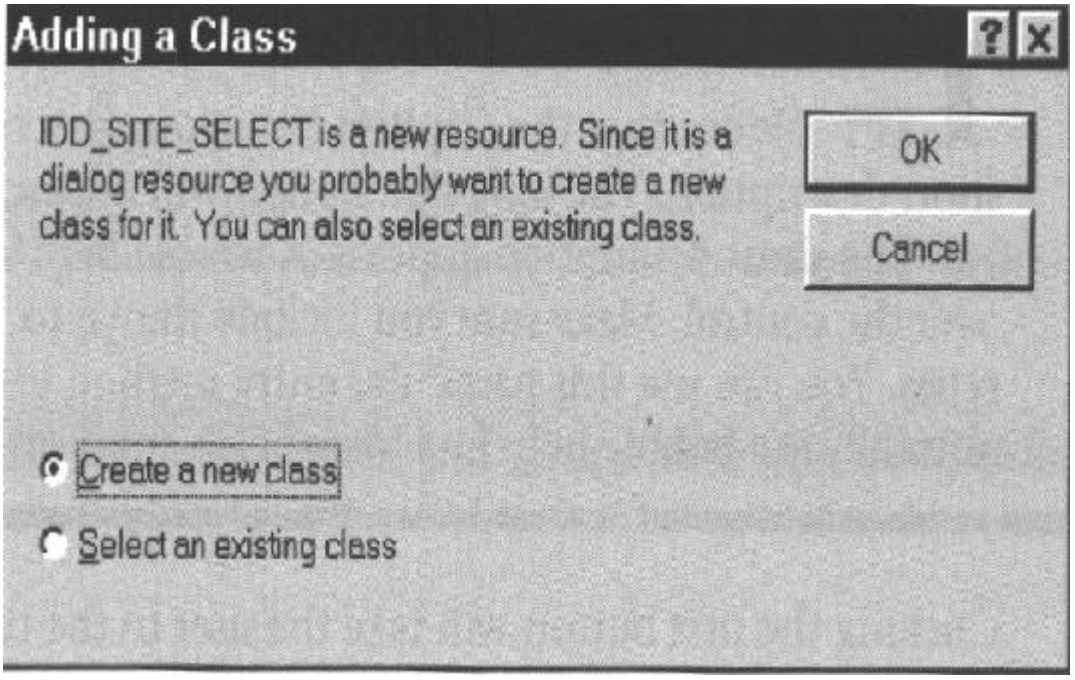


图 12.2　如果用户选择访问主 Web 站点之外的地方，示例程序将显示该对话框

双击每个编辑控件显示它们的属性对话框，设置一个易于记忆的 ID。第一个编辑控件的 ID 为 IDC_URL，第二个编辑控件的 ID 为 IDC_ANCHOR_NAME，第三个编辑控件的 ID 为 IDC_FRAME_NAME。稍后将会介绍这些名称的作用。

定义新类并编写代码

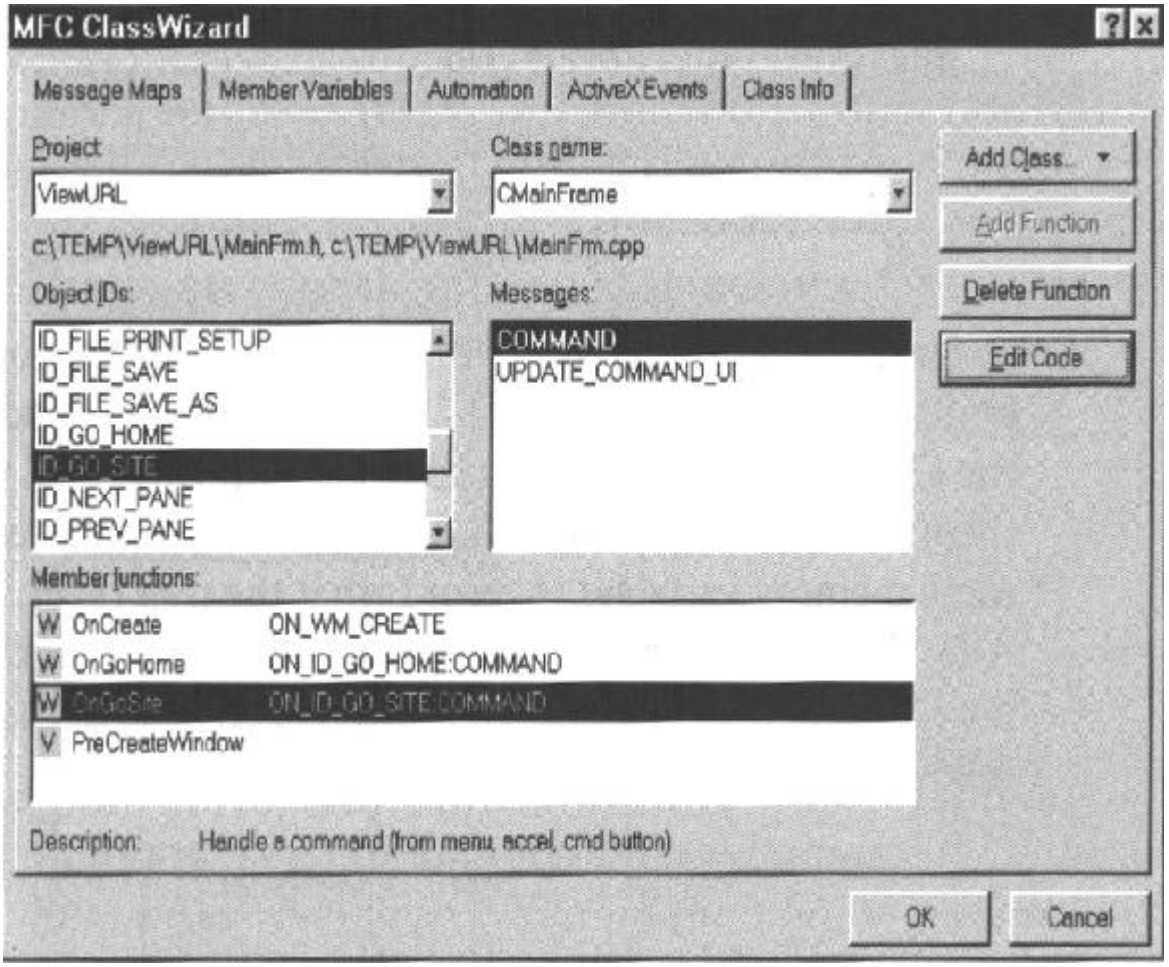
现在已经定义了所有所需的资源，应该给添加到工具栏中的两个按钮增加代码了。确保已经选择了 IDD_SITE_SELECT 对话框，然后用 View|ClassWizard 命令显示 ClassWizard 对话框。在这种情况下，你将看到如下图所示的对话框，它表明 IDD_SITE_SELECT 是一个新对话框，需要为它创建一个类。



单击 OK，你将看到 New Class 对话框。你需要提供的只是类名。示例程序使用的是 CSiteSelect，它使得类名及其相关资源很易于识别。本例使用的是缺省的 CDialog 基类。单击 OK 完成操作。

选择 ClassWizard 的 Message Map 页，然后从 Class Name 字段中选择 CMainFrame 类名。滚动下拉对话框左边的 Object IDs 直至找到 ID_GO_HOME。

单击该项，然后单击对话框右边的 Message 字段中的 COMMAND 项。单击 Add function 给应用程序添加所需函数。此时会看到 Add Member Function 对话框。单击 OK 接受缺省函数名。同样处理 ID_GO_SITE 对象标识符。完成之后，MFC ClassWizard 对话框如下图所示。



选择 OnGoHome 项并单击 Edit Code。MFC ClassWizard 就会直接进入新函数框架。程序列表 12.1 显示了该按钮的代码，确保将 HlinkSimpleNavigateToString()函数调用的第一个参数与 Web 服务器的缺省页面

地址相匹配。你会看到 OnGoSite()函数就在 OnGoHome() 函数之后。给它添加程序列表 12.2 所示的代码。确保在 MainFrm.Cpp 文件开头包括了 SiteSelect.H（如果愿意的话，还可在 STDAFX.H 文件中包含它）。

程序列表 12.1

```
void CMainFrame::OnGoHome()
{
        // Go right to the company's home page.
        HLindSimpleNavigateToString(L"http://aux/default.htm",
                NULL,NULL,NULL,0,NULL,NULL,0);
}
```

程序列表 12.2

```
void CMainframe::OnGoSite()
{
        //Create a copy of the dialog.
        CSiteSelect    NewSiteSelect;

        //Display it.
        NewSiteSelect.DoModal();
}
```

现在已有足够的代码可以将用户送往主页（程序列表 12.1）或显示对话框（程序列表 12.2）上了。不过还需要给 CSiteSelect 对话框增加一些代码。当用户单击 OK 时，你可能想把他们带到他们选择的其它站点。首先要添加一些内存变量来记录用户的输入。再来看一看 IDD_SITE_SELECT 对话框。按下 CTRL 键并双击第一个编辑框，就能看到如下图所示的 Add Member Variable 对话框。

它允许你给 IDC_URL 编辑框赋予一个变量。本例中命名为 URL（该名字将会在对话框中以 m_URL 出现）。选择 Category 域中的 Control，把 Variable Type 域改变成 CEdit ，CEdit 就是该例所需的变量类型。单击 OK 完成操作。同样处理另两个编辑框，并将第二个变量命名为 Anchor，第三个命名为 Frame。

有了变量，现在还要添加另一个函数，用 View|ClassWizard 命令打开 MFC ClassWizard。对话框应该仍然显示 Message Map 页； 否则的话就选中该页。在 Class Name 域中选择 CSiteSelect 项，就会看到对话框左边的 Object IDs 列表变为 CSiteSelect 对话框提供的内容。选中 IDOK 对象标识符和 Message 域中的 BN_CLICKED 项。单击 Add Function，然后单击 Add Member Function 对话框中的 OK。这时 MFC ClassWizard 对话框如下图所示。

现在，单击 Edit Code 显示函数的框架。该函数接收用户在对话框中的输入信息，然后将它们带到 Internet 上的特定位置。程序列表 12.3 显示了完成该任务的代码。

程序列表 12.3

```
void CSiteSelect::OnOk()
{
```

```
LPCWSTR     pszURL = NULL;       //local copy of URL string.
LPCWSTR     pszAnchor = NULL;    //local copy of Anchor string.
LPCWSTR     pszFrame = NULL;     //local copy of Frame string.
LPTSTR      pszBuffer = "";      //Conversion buffer.

// Get the URL string and convert it .
if (m_URL.LineLength(0) > 0)
{
        m_URL.GetLine(0,pszBuffer,m_URL.LineLength(0));
        pszURL = LPCWSTR(pszBuffer);
}
else
{
        MessageBox("You must enter a URL","Error",MB_OK);
        return;
}

//Get the Anchor string and convert it.
if (m_Anchor.LineLength(0) > 0)
{
        m_Anchor.GetLine(0,pszBuffer,m_Anchor.LineLength(0));
```

```
            pszAnchor = LPCWSTR(pszBuffer);
        }

        //Get the Frame string and convert it.
        if (m_Frame.LineLength(0) > 0)
        {
            m_Frame.GetLine(0,pszBuffer,m_Frame.LineLength(0));
            pszFrame = LPCWSTR(pxzBuffer);
        }

        //Go right to the selected page.
        HLinkSimpleNavigateToString(pszURL,
            pszAnchor, pszFrame, NULL, 0, NULL,NULL,0);
        CDialog::OnOK();
    }
```

上述代码中有一些小技巧，但大部分 C **语言程序员使用过这些技巧。首先，要将三个用于保存位置信息的变量置为** NULL**。这样，即使它们没有什么信息可保存，也能正常运行。这时，代码仍要强制用户提供** URL**，只有一种情况不必这么做，即，如果你允许用户跳到他们当前所在页的一个锚地，就可以让** URL **参数仍然是** NULL **并简单地让它们指定锚地名称。保存对话框中信息的方法需要使用缓冲变量。但从** LPTSTR **（指向** Windows **或** Unicode **的以** NULL **结尾的**

长指针）到 LPCWSTR（指向以 NULL 结尾的 Unicode 串常量的长指针）的转换很简单。这个特殊例程的核心是调用 HLinkSimpleNavigateToString，个函数负责将位置信息转换为真正的 URL。

现在，让我们看一下应用程序是什么样子。和通常一样编译并运行应用程序。如果单击 Home Web Page 按钮，就能看到 Internet Explorer（或其它缺省浏览器）启动了，并把你带到设为主页的任何 Internet 站点。图 12.3 显示了本例按下该按钮的结果（图中显示的主页是作者的 Internet 服务器设置，你的屏幕上应该显示自己的设置）。一定要将程序列表 12.1 中 HLinkSimpleNavigateToString() 调用的地址设置为自己的主 Web 站点，否则就会出现意外结果。

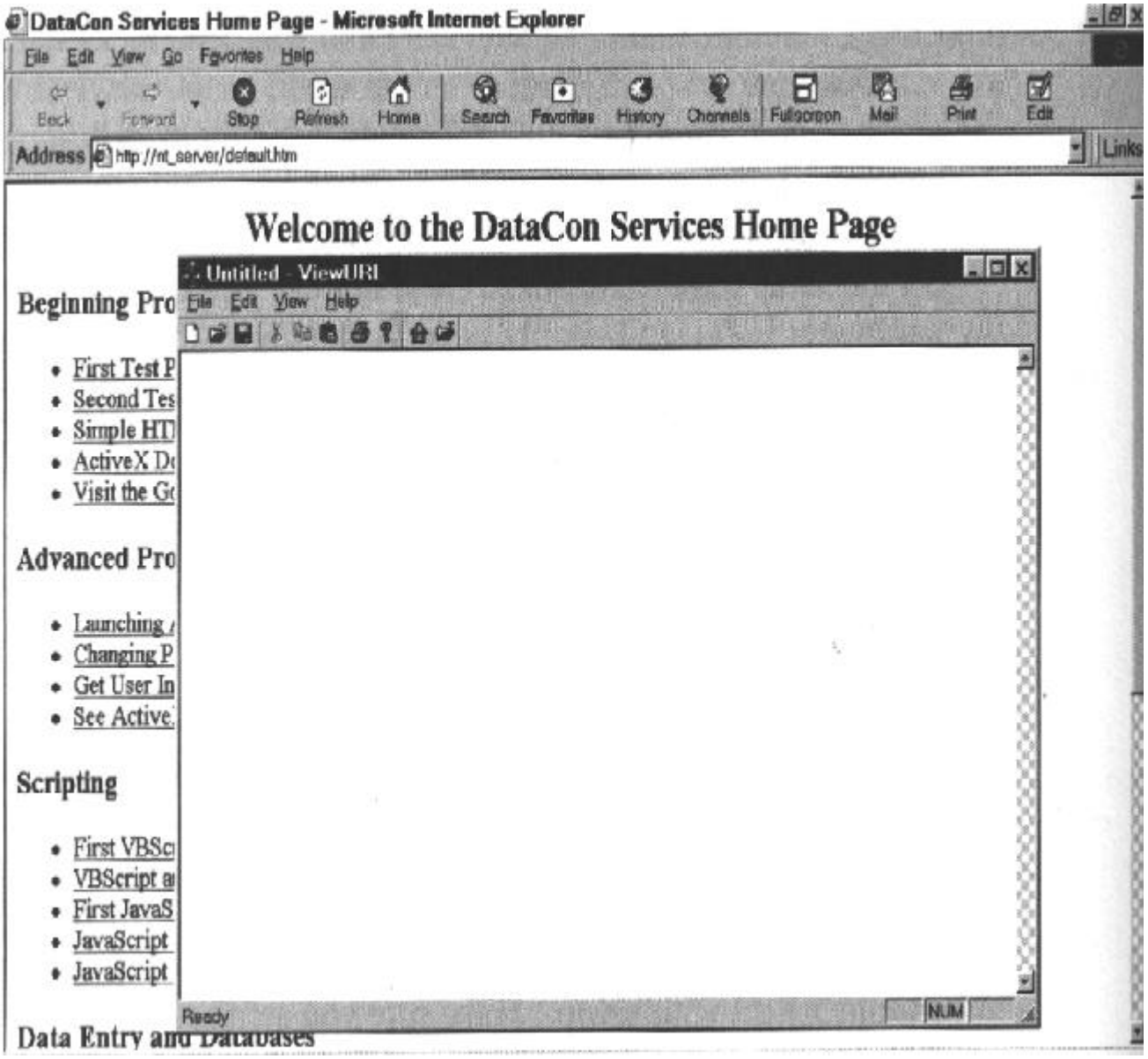
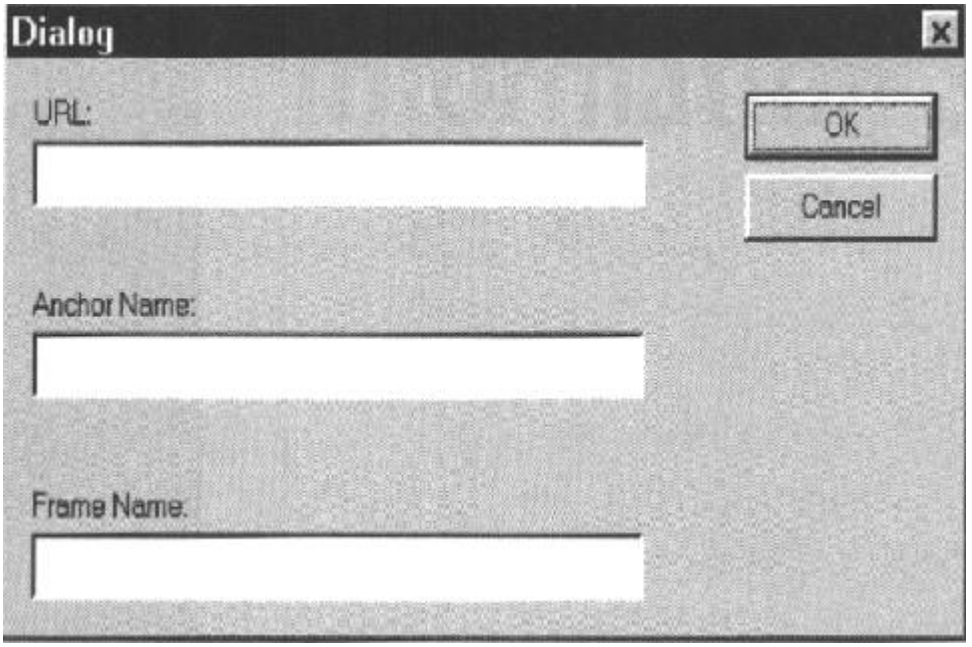


图 12.3　**可以给任何应用程序添加这样的按钮，从而允许用户很容易地访问** Internet

单击 Any Web Page **按钮会显示如下图所示的对话框。**



它允许输入一个 URL**、可选的锚地名或框架名。目标** Web **页必须支持所指定的锚地名和框架名，否则函数调用会失败。如果输入本地测试机器支持的地址之外的** URL **值，那么应用程序还需要一个活动的** Internet **连接。**

第 13 章　使用 Internet 信息服务器(IIS)

在工作中有许多种不同的方法使用 IIS(Internet Information Server)，其中大部分都是通过传统的静态 HTML 方案，它需要使用各种类型的脚本。例如，可以在 C/C++ 程序中使用 CGI （公共网关接口）脚本查询一个数据库。脚本将会形成一个包含查询结果的 HTML 页并将它返回给客户机。本章将不再介绍这些传统的方法，因为这方面的内容已经有许多书籍介绍过了。需要着重指出的一点是，传统方法之所以仍然存在，一方面是因为它们确实能够有效地工作，另一方面，当我们本章讨论的新技术还没来得及担当重任时，仍然需要使用旧方法。例如，你可能已有一个现存的基础结构，那么试着用新技术对它彻底重新编程就可能不合算。

WEB 链接　有一种很快捷的方式可供学习包括 ODBC 在内的某些 IIS 访问技术的基础知识。Dynamic Systems Internaitonal（国际动态系统）提供了一系列有关 IIS 的课程，你可以从 http://www.dsi.org/dsi/iis.htm 下载它们，这些课程包括了使一些旧的技术访问方法能进行工作的全部内容的概述。你还可以访问 Microsoft IIS FAQ 站点（由 Stephen Genusa 负责主办）。该站点位于 http://www.genusa.com/iis/，它包括了一些像 CGI 这样的论

题。另外，位于 http://www.Adiscon.com/IIS 的 KLV 站点也包括了一些关于在使用老技术时的 Visual C++编程中各种注意事项的链接，这个站点还包括一些其它链接，据此可以找到有关 Internet 制定规范的活动以及安全问题（如用户认证等）的最新信息。

另一方面，使用像 Active Server Pages(ASP)这样的新技术还能提供更大的灵活性并极大地提高你在 Internet 上提供内容的能力。这就是本章要介绍的内容——可用来在自己的 Web 站点上提供更好内容的新技术。另外本章还将介绍一种给 Web 站点用户提供更大灵活性的特殊方法。本章的主要内容就是介绍使用 Internet 服务器应用程序设计接口(ISAPI)的方法，另外还要介绍与 ISAPI 相关的一些支持技术，比如 ISA（ISAPI 服务器应用程序）。学完本章后，对这种技术在现实世界中的工作会从总体上有个很好的了解，清楚自己在什么时候需要使用那些老技术。另外，我肯定要把在自己的服务器上实现这些新技术时应注意的一些问题介绍一下给你。知道这些问题将有助于你决定使用新技术还是使用以前用过的旧方法。

那么什么是 ISAPI 呢? 就本书而言，ISAPI 是一系列 MFC 扩展，它允许你直接使用 IIS。本章将使用一种新型工程来实现 ISAPI，即 ISAPI Extension Wizard（ISAPI 扩展向导）。另外还将介绍五个新类：CHttpSever、CHttpServerContext、CHttpFilter 、CHttpFilterContext 和 CHtmlStream。我们将使用这些类来创建 ISA——它还有许多其它名称，例如在 Microsoft 文档中就被称为 ISAPI 服务器扩展 DLL。为简单起见本书将使用 ISA 这个名称。

注 ISAPI 是允许你在工作中使用 IIS 的一系列特殊 MFC 扩展。

ISA 必然依赖于 ISAPI。你可以用 ISAPI 类来创建 ISA 扩展以及 IIS 的过滤器。不过并不仅限于 ISAPI 类，还有用于控制 Internet 通信的 WinInet 类以及提供像接口这些东西的所有标准 MFC 类。过滤器允许你通过监测服务器上的事件决定是让它进入还是将它拒之门外。例如，可以创建一个 ISA 过滤器，除非用户输入了正确的口令，否则，将被 Web 站点拒之门外。另一种类型的过滤器，对于文件规模大于某值的文件，拒绝将它装入 Web 站点上的 FTP 服务器。扩展更像是应用程序或后台进程。例如，可以创建一个扩展允许用户不必使用脚本就能与数据库进行交互。同一个扩展可以根据用户的输入和服务器上数据库的内容，动态创建 Web 页。

注　ISA 既可以对信息访问进行过滤，也能扩展 IIS 提供的服务。

在阅读本章前，还应回顾一下第 8 章到第 12 章中讨论的关于客户机的信息。第 8 章帮助你理解 HTML 的基础，第 9 章帮助你理解 JavaScript，第 10 章介绍 ActiveX 控件编程的基础知识，第 11 章帮助你理解 ActiveX Document 的编程，而第 12 章则讨论了非常重要的 URL moniker。所有这些章节都能帮助你了解客户机对服务器的需求——如果你计划修改一下服务器，那么这些信息都是重要信息。第 12 章特别重要，因为它能帮助你理解客户机与服务器之间的连接——即进行对话的支持机制。

注释 本章的代码都是使用 Visual C++ 6.0 开发的，并且没在老平台上进行测试（在 C++ 5.0 中肯定没问题）。尽管这些代码可能不经修改就可以在 Visual C++ 4.2 下工作，但你应在我提供的过程中加以考虑。代码本身如果要在 4.2 版以前的 Visual C++ 上编译，需要进行某种修改。

下面各节将逐一介绍 ISAPI 编程的五个重要方面。首先是 ISAPI 自身简介。这是理论部分，如果你已经对 ISAPI 的工作原理很清楚，并想直接开始创建自己的程序，就可跳过这一节。本书特意将有关过滤器和扩展的创建与它们的实现分开论述，以便更通用。而且创建一个 ISA 与实现它是不同的步骤。本章第二部分介绍如何创建 ISAPI 扩展，第三部分介绍如何在服务器上实现它。类似地，第四部分介绍如何创建 ISAPI 过滤器，第五部分是如何实现 ISAPI 过滤器。

本章最后两节介绍怎样创建其他类型的 ISAPI 扩展和过滤器。第 6 节提供了另一个 ISAPI 扩展的程序设计例子，而第 7 节则提供了第二个 ISAPI 过滤器程序设计的例子。这些例子使你更好地掌握怎样使用 ISAPI 来增强服务器的操作性能、向用户提供更好的输出或者使你的网络更加安全。

WEB 链接 如果你遇到有关 IIS、ASP 或 ISA 的问题，可以访问许多新闻组以获得帮助。实际中的问题太多了，本章不可能将它们一一列出，所以你还是花点时间自己去找吧。要得到 Microsoft 对 IIS 专门的最佳支持，可参阅 microsoft.public.inetserver.iis 。在 microsoft.public.inetserver 区还有其它一些新闻组，不过刚才说的新闻组消息最多。你还可以在

microsoft.public.inetserver.iis.activeServerPages 找到 Microsoft 专用的 ASP 帮助。另一个更有趣的非 Microsoft 站点是 comp.lang.java，在该站点上有关于从 ASP 到 ActiveX 的各种事情的消息线索，这一点真让我感到惊奇。另一个很好的有关 IIS 专用帮助的非 Microsoft 站点是 comp.infosystems.www.servers.ms-windows。在写本书时作者就在该新闻组中找到一条非常棒的 ISAPI 信息。毫无疑问还有其它 comp.infosystems.www 新闻组可以查看。如果你使用 FrontPage 作为自己 Web 页维护工具之一，就应看看有关 ISAPI 专用帮助的 microsoft.public.frontpage.client。通常要想找出软件中的错误是非常困难的，但是如果你看看 comp.os.ms-windows.nt.software.compatibility，那么就会发现 IIS 并不困难。最后，如果你想检索一些隐藏的 ASP 新闻组，可以访问 microsoft.public.activex .programming.scripting.vbscript。

13.1 ISAPI 概述

了解一下 ISAPI(Internet Server API)在 Microsoft 产品中处于什么位置，对于理解它本身不无帮助。实际上 Visual C++提供了五个层次的支持，其中三个层次位于服务器上。另外两个层次支持是客户机专用的——你永远不会在服务器上看到它们。下面将逐一定义各个层次支持并告诉你到哪里去找到它们。

- ISAPI（服务器） 这是本章要讨论的支持层。你需要给服务器本身提供一个扩展或过滤器。也就是说，客户机不会直接与该层次支持打交道。它只能看到交互的结果。
- WinInet（服务器或客户机） 本书不直接涉及该支持层次，但确实会间接地涉及到它。这些类允许你在客户机和服务器之间使用专用的数据传送方法。其中包含三种协议支持：HTTP、FTP 和 Gopher。从本质上看，你需要用这些类来创建一个会话(CInternetSession)，它是服务器的一个连接，然后指定连接类型(CFtpConnection 或者 CHttpConnection)。建立连接后，你就可以进行工作了，比如查找文件等（CFtpFileFind 或 GopherFileFind）。通常你不必直接与这些类打交道，因为 Visual C++已经做好了一切，就像第 12 章创建 URL moniker 应用程序时一样。
- 异步 URL moniker(服务器或客户机） 第 12 章已经介绍了 Visual C++ 的 Internet 支持的特定领域。要记住的是异步 URL moniker 允许你在 Internet 上完成任务而不必等待。只要简单地告诉目标应用程序你想要什么，然后就可以干其它事情了。这里的主要思想是，大部分情况下 Internet 不提供即时响应，如果不使用异步 URL moniker 方式，那么对于长时间的下载，会使用户的机器长时间地等待而无所事事。
- ActiveX document（客户机） 该层次的支持在浏览器中显示文档并允许用户编辑它。这些内容第 11 章都已经介绍过了。
- ActiveX 控件（客户机） 过去创建一个 Web 页的基本元素要涉及到编写大量的脚本，即使那样，得到的也只是静态的画面。在 Web 站点上

使用 ActiveX 控件意味着老的技术和静态显示再也不拖你的后腿了——现在的Web 页可以根据具体情况发生变化。第 10 章介绍的就是该技术。

Web 链接 Internet 上有许多地方可以获得 IIS 、 ISAPI 以及其它与 Internet 服务器有关的技术（如 Active Server Page(ASP) ）。访问 IIS 信息的主要站点是 http: //www.microsoft.com/ntserver/Basics/WebServices/default.asp 。Microsoft 支持的该站点包括了一些链接，它们指向你使用 IIS 本身所需的一切知识，甚至还有一些公用链接指向使能(Enabling)技术，（例如 ISAPI）。如果你想获得非 Microsoft 的帮助，可以访问位于 http://www.genusa.com/asp/的 ASP 开发者站点。它包含许多关于在 Internet 服务器上使用 Active Server Pages 的有价值的信息，另外它还有非 Microsoft 链接指向的 IIS 和 ISAPI 站点。

在现实世界中使用 ISAPI

在现实世界产品环境中，你需要了解一些与 ISAPI 有关的事情。例如，你的ISA 是要装入服务器的一个 DLL，这与其它 DLL 完全一样。该 DLL 将与 HTTP 服务器共享同一地址空间，如果需要把该 DLL 占用的内存用于其它用途，还可以卸载这个 DLL。正是由于这一点，ISAPI 才赢得了你的青睐。与其它技术（如 CGI）比较起来，使用 ISA 有诸多好处，下面逐一予以介绍。

◆ **内存占用少** 由于 ISA 是作为 DLL 装在服务器上的，并和 HTTP 服务

器共用同一内存空间，所以不必浪费那些使用 CGI 时用到的内存。你真正要装载的只是要求 ISA 完成任务所需的业务逻辑。

- 速度　第一次装载 DLL 与第一次装载 C 应用程序所花费的时间基本相同，尽管 DLL 因为较小可能要快一些。不过，一旦第一次装载了 DLL 后，它就常驻内存直至卸载它，这就意味着可以不必再花费装载时间。而 CGI 脚本是每次调用时都装载一次 C 应用程序。这可不是好消息。因为 ISA DLL 与 HTTP 服务器共享同一内存空间，就不会为进程间调用浪费时间了，也不会遇到在不同地址空间中使用 C 应用程序时遇到的其它时间耗费问题。
- 代码共享　服务器的任务就是一次性装入你的 DLL。申请 DLL 服务的任意应用程序都可以访问它。显然，代码共享是导致前两点中提到的对 CGI 进行改进来获得低内存消耗和提高速度的因素之一，而且，代码共享还带来不那么明显看到的益处。例如，代码共享减少了服务器的管理时间，因为管理员只需要对给定的 DLL 再产生一个副本去影响使用它的每一个应用程序。CGI 通常使用的 C 应用程序有大量的冗余代码，改变一个单一例程，就需要对服务器上使用该例程的每一个 C 应用程序进行改变。这就意味着增加管理员的时间和程序员追踪附加应用程序的需求。
- 可靠性　CGI 脚本使用的 C 应用程序在装入服务器并在服务器上执行时，并没有对服务器本身作多少访问。因此，就难以创建一个 C 应用程序来监督服务器事件并从出错中恢复。通常服务器总是终止出了错的

CGI，而客户机却什么也不知道。ISA 拥有对服务器的完全访问，这就是说它们可以更容易地从出错中恢复。于是，客户机二次申请服务器的情况（即使有的话）非常少见。

- 过滤能力　使用 CGI 和 C 应用程序，你不能提供与 ISA 一样的事件驱动功能。原因很简单：C 应用程序是被调用、做工作、然后卸载。所以它没有办法长时间一直监视服务器事件。
- 在一个 DLL 中的多任务能力　要 CGI 完成的每个任务都需要你有完成每个任务的单独的可执行程序。结果导致过度调用每个例程。而 ISA 却能包含多个命令，每个命令是 CHttpServer 类的一员。

获得这六项功能并不意味着在学习或写出代码方面付出昂贵代价。ISA 就像 CGI 一样好用。下面两行代码看上去就像是在应用程序中的代码一样。

```
<!-这是带有一个参数 CGI 例程的调用。->
http://aux/controls/sample.exe?Param

<!-这是带有一个参数 ISA 例程的调用。->
http://aux/contorls/sample.dll?Param
```

可以看出操作 ISAPI 一点也不难。我们使用了与调用 CGI 例程几乎完全一样的代码来调用 ISA 控件。实际上，从编码角度来看，唯一的不同就是 ISA 控件使用 DLL 扩展名，而 CGI 例程使用 EXE 扩展名。从理论上来说，你可以将服务器转换为 ISA，对 Web 页面作些检索替换性的修改，这从用户界面角度几乎看不出什么区别。当然，每个人都会注意到，因为使用 ISAPI 会带来 Web

站点的更高效率。

上面我提到的使用 ISA 的最后一个好处是，你可以用一个 ISA 完成多个任务。稍后将会介绍如何实现这种机制。现在你只需要知道，调用语法与以前调用标准 CGI 没什么太多不同。下面就是一个 ISA 例程调用，它指明了你要使用 DisplayStr 函数。

```
<!- 用一个参数在一个 ISA 例程中调用 DisplayStr 函数。->
http://aux/controls/sample.DLL?DisplayStr?Param
```

可以看出，在调用串中，我们使用第二个问号(?)调用了缺省任务之外的某种东西。第一个参数告诉服务器在 SAMPLE.DLL 中要调用哪个函数，第二个参数则包含一系列参数。这种调用函数的方法称为分析映射，本章“创建 ISAPI 扩展”一节将对如何创建它们作进一步介绍。

注 给调用语法再加第二个问号就可以在 ISA 中调用缺省函数之外的东西。

ISA 确实还有一些与 CGI 相同的特性。例如，应用程序是在服务器而不是客户机上运行的。这就使升级应用程序变得很容易——你要做的工作只是在服务器上更换一个 DLL 文件（一定要终止服务才能更新 DLL，不过这是方便地对机器进行更新所需付出的最小代价）。显然，这比在访问 Web 站点的每个客户机上都替换一个应用程序要容易多了。这也正是目前各家公司都开始重视把内部网作为承担像帮助系统和定制数据库应用程序这样的主要任务的原因之一：更新一个服务器比更新许多客户机要容易得多。

注 ISA 在服务器上运行，意味着每次更新一个定制应用程序时只需更新服务器而不用更新各个客户机。

技巧 IIS 第 4 版添加了一些新功能，使得 ISA 更容易使用。例如，可以让服务器在每次调用之后就卸载 ISA。这意味着你能试试 ISA，看它能否工作，如有必要，用新版本去替换一下，而这一切都是在不停止服务的条件下进行的。这一新功能的不足之处是，由于每当客户机调用 ISA 时，服务器都需要重载它，这样，就带来一些操作问题。

在过滤器与扩展之间进行抉择

你不必花太多时间就可以决定出是创建 ISA 过滤器还是创建 ISA 扩展（或者两个都创建）。这两种 ISA 之间的区别很明显。知道它们的差别后选择所需的 ISA 就是件简单的事了。

过滤器总是对服务器本身的事件作出反应。如果用户试图登录 Web 站点，就会生成 ISA 过滤器能监测到的一个事件。通常用过滤器来修改客户机与服务器之间的数据流。例如，如果应用程序发现有 SF_NOTIFY_AUTHENTICATION 事件发生，就可以显示一个对话框，要求用户输入名称和口令。如果名称和口令都正确，用户即可获得对 Web 站点的访问权。

扩展适用于 CGI 同样的情形。例如，如果想创建一个订货系统，就可以使用扩展。扩展将接收用户填入窗体中的数据，处理信息并将它们加入数据库，最后将某种形式的收据发还给用户。这正是使用 CGI 脚本时遵循的同一过程；唯一的区别是现在使用的是 DLL。与过滤器不同，扩展不监测服务器上的事件，无论从哪一方面看它都像应用程序那样工作。

还有一些情况，不管选择过滤器还是扩展都没什么关系。例如，考虑一个简单的 Web 计数器。每次有人访问 Web 站点时，都更新计数器并显示它们是第几位访问者。如果愿意的话，你可以从 Web 页面上调用 ISA 扩展来更新计数器，也可以用过滤器监测服务器上的 SF_NOTIFY_LOG 事件并自动发送信息。这只是个人喜好问题。

注 像访问计数器这类的 ISA 应用程序，我们既可以选择创建过滤器，也可选择创建扩展。

你可能还会遇到需要同时使用扩展和过滤器的情形。例如，你可能想为授权的 Netscape 用户显示一种 Web 页，而给非授权 Netscape 用户显示另一种 Web 页。针对授权的和非授权的 Internet Explorer 用户也有两个 Web 页。这种情况下需要在 HTML 格式中编写一些脚本来检测浏览器类型，需要用过滤器来检查用户是否授权，还需要用扩展来生成正确的页面。在服务器中将客户机脚本和 ISA 扩展进行组合，将大大丰富你的 Web 站点并使它便于每个人使用。

注 某些 ISA 应用程序（如专用的浏览器支持），要求创建既包括过滤器又包括扩展的复杂的 DLL。

使用 ISAPI 的五个类

前面已经说过本章将使用五个新的 MFC 类。这并不是说你不再使用那些以前已经熟悉的类了，不过这些类确实能提供设计 ISA 过滤器或扩展所需的特殊功能。下面将对下面章节中要用的这些类作简单的介绍。显然在开始设计真正的示例代码时你的了解会更深入一些的。

- CHttpServer　这是创建过滤器或扩展所需的主要类。CHttpServer 类定义了一个作为服务器扩展 DLL 或 ISA 的对象。该 CHttpServer 对象的主要目的是处理由服务器和客户机生成的消息。所以任何 ISA DLL 中只有一个 CHttpServer 对象。
- CHttpServerContext　CHttpServer 真正创建了该对象，它是来自客户机的某个单一请求的单一实例。也就是说，对每个活动的客户机请求，都有一个 CHttpServerContext 对象。当客户机请求完成后该对象即被清除。
- CHttpFilter　ISA 过滤器要求使用特殊的 CHttpFilter 对象来管理服务器的事件。该对象负责查找所有客户机与服务器交互时由服务器生成的 SF 消息。例如，每次用户试图访问 Web 站点时，客户机都会生成一个 SF_NOTIFY_LOG 消息。
- CHttpFilterContext　CHttpFilter 对象将为每个活动的服务器事件创建一个 CHttpFilterContext 对象。CHttpFilterContext 类表示在特定会话中由单个客户机形成的单个事件。
- CHtmlStream　利用该类来管理客户机和服务器间的数据。无论何时当 CHttpServer 类需要在客户机和服务器之间传递信息时，都会创建该类的对象。大部分情况下每个 DLL 只有一个这样的对象。不过，CHttpServer 对象为了安全传递数据可以创建所需的任意多的 CHtmlStream 对象。典型的 CHtmlStream 对象包含创建 HTML 页内容所需的数据和所有标记。例如，如果你是检索数据库，那么结果的 CHtmlStream 对象就不仅包含数据，还包括实际创建的用来显示数据 Web 页所需的 HTML 标记（参

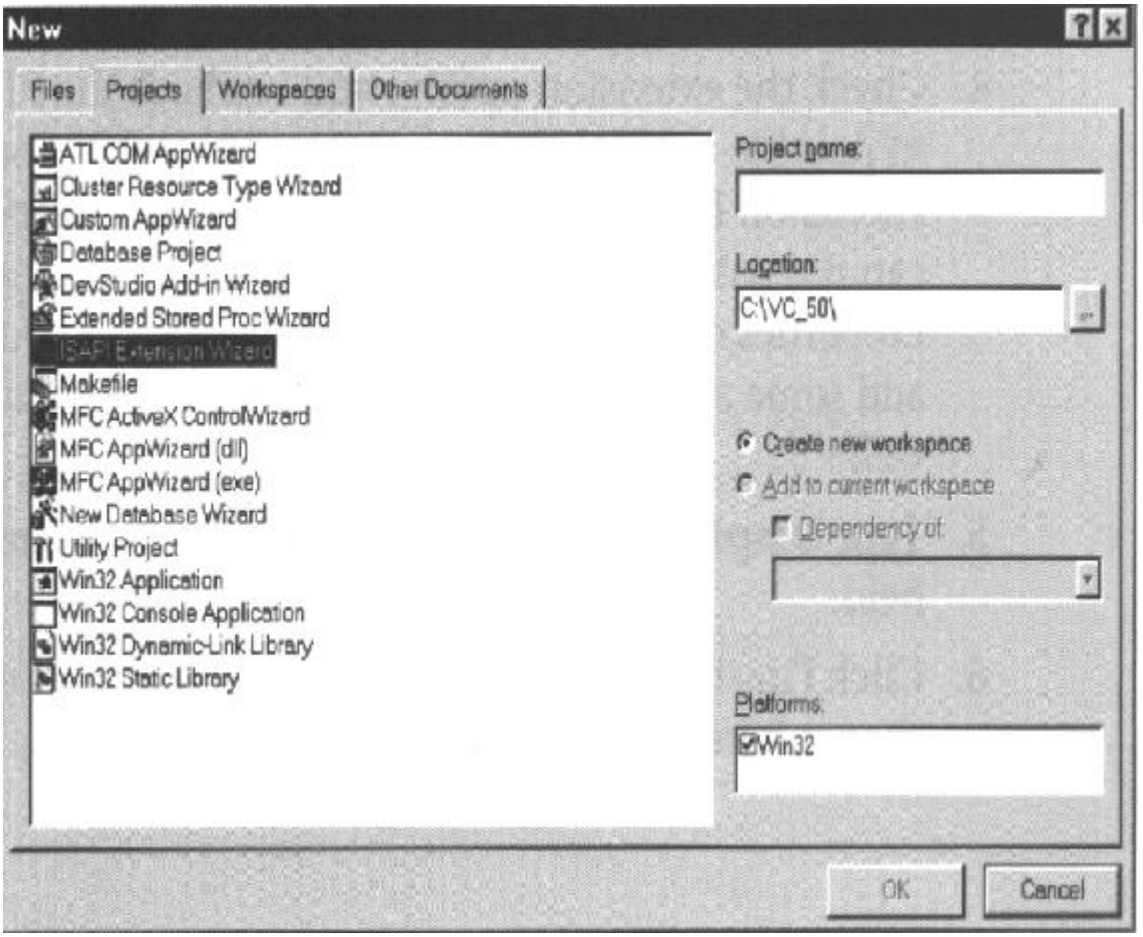
见第 8 章有关 HTML 标记的介绍）。

13.2 创建 ISAPI 扩展

好了，现在让我们看一下我们的第一个 ISA。这个例子中将利用 ISAPI Extension Wizard（ISAPI 扩展向导）来创建一个简单的 ISAPI 扩展。全部目的在于让你熟悉概念并介绍一些使用 ISAPI 的技术。例子程序接收来自客户机的一个串，稍作修改，然后用一个新页显示客户机发送来的这个串。下面介绍的过程将带着你一步步地创建示例 ISA。你可以使用同样的过程来做需要创建的任何 ISA 扩展。我们假定你已经启动 Visual C++ 并且使用的是 6.0 版。

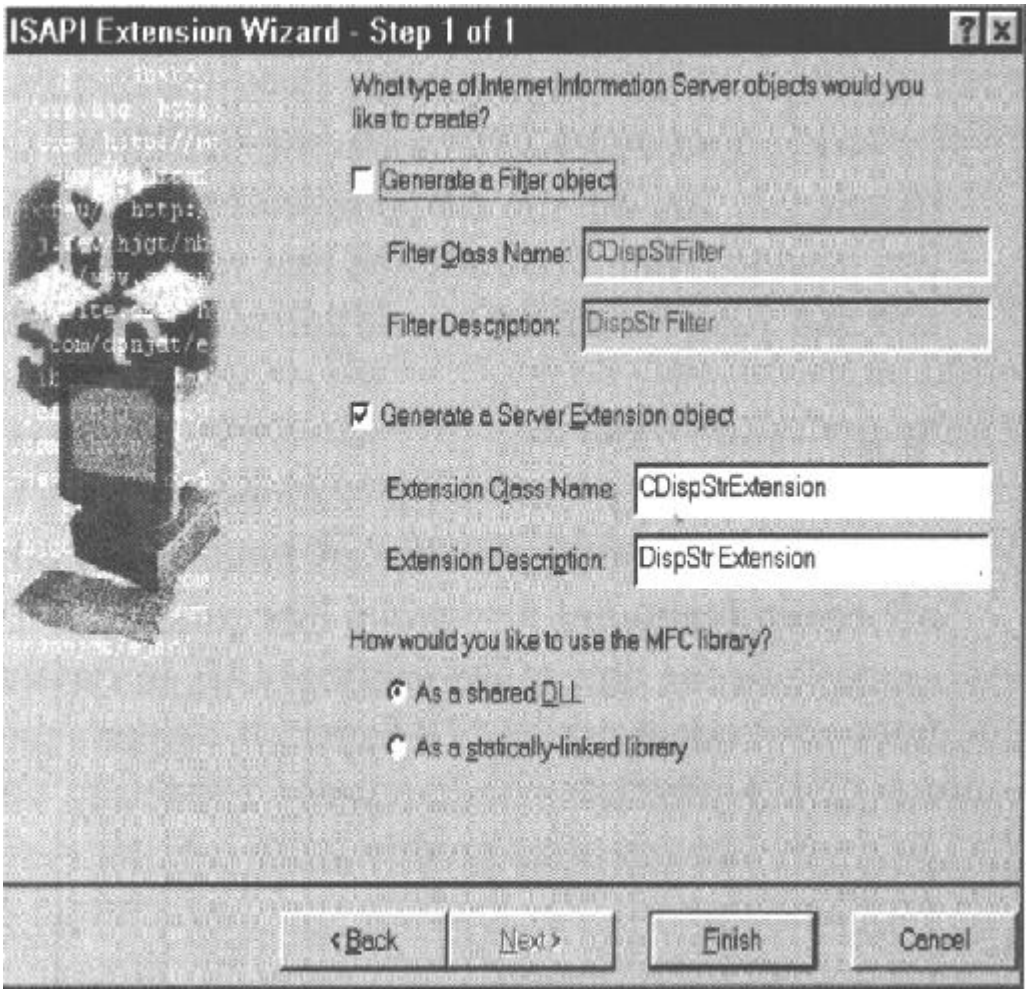
WEB 链接 在创建自己的 ISAPI 扩展或过滤器时，最好的资源就是其它程序员编写的免费软件。alt.comp.freeware 新闻组中列出了相当多的这样的软件。例如，在写本书的时候，AAIT Incorporated 就推出了一个新的免费软件，称为 CGI Expert（CGI 专家）。它同时支持 CGI、Win-CGI、ISAPI 以及 NSAPI 接口。显然谁都想用免费产品，但是就像使用其它产品一样，一定要小心。不过它们确实对如何创建自己的定制扩展提供了极好的想法（或者可以提供一种解决方案使得你自己不必去编程了）。

1. 使用 File（文件）|New（新建）命令显示如下图所示的 New（新建）对话框。请注意，我已经选择了 ISAPI Extension Wizard（ISAPI 扩展向导）。



2. **在工程名称域中键入想创建的** ISA **的名字。这个例子中使用的是** DispStr，**不过可以使用你想使用的任何名称。**

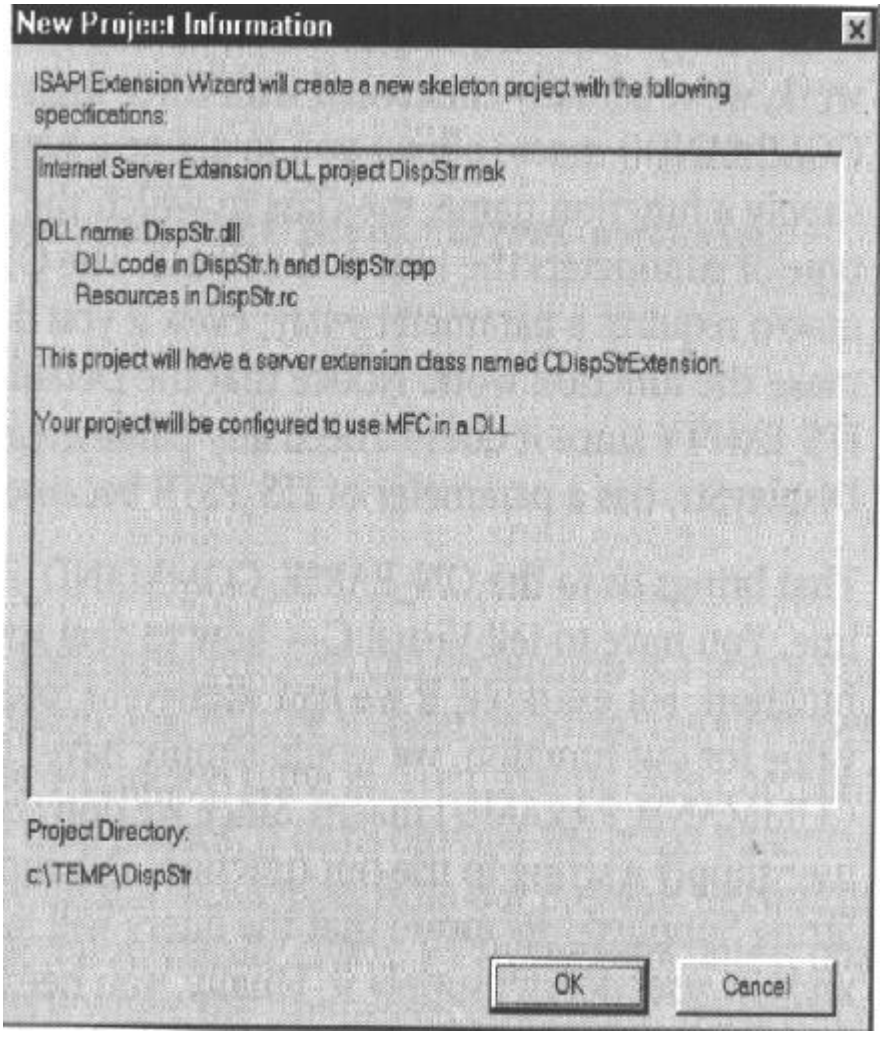
3. **单击** OK，**就会看到如下图所示的** ISAPI Extension Wizard（ISAPI **扩展向导**）Step 1 of 1（**第一步**）**对话框。在这里可以为你的** ISA **选择各种特性。注意对话框中有三个主要区域。选中第一个复选框，就可以创建过滤器；选中第二个复选框，就可以创建扩展。第三个区域定义如何将** MFC **与你的应用程序链接起来。**



4. **选中扩展选项，而不是过滤器选项。在** Extension Description **（扩展描述）域中，你应对你的** ISA **提供简短扼要的说明。该说明将以一个字符串的形式出现，你可在** DLL **中需要时使用它。当某人打开你的** ISA **的** Properties(**属性**)**对话框时，该说明并不出现在对话框中，所以需要给** DLL **的版本信息再添加点附加文本。**

5. **在** Extension Description **（扩展描述）域输入** Display a string from the client **（显示来自客户机的串）。**

6. **单击** Finish（**完成**）**就会看到如下图所示的** New Project Information （**新建工程信息**）**对话框。**



7. **单击** OK，ISAPI Extension Wizard（ISAPI **扩展向导**）**就完成了所需的程序框架的创建。**

现在，为了使扩展具有实用功能，还要对 ISAPI Extension Wizard（ISAPI **扩展向导**）**创建的框架做三件事。首先是为添加的新函数创建一个分析映射**（ Parse

map)。在 DispStr.cpp 文件的顶部可以找到该分析映射。程序列表 13.1 列出了应该给该分析映射加入的代码，以便我们能够从 HTML 页面访问这个新增函数。

程序列表 13.1

```
//////////////////////////////////////////////////////////////////////
// command-parsing map

BEGIN_PARSE_MAP(CDispStrExtension,CHttpServer)
    //Our special DisplayStr parse command.
    ON_PARSE_COMMAND(DisplayStr,CDispStrExtension,ITS_PSTR)
    ON_PARSE_COMMAND_PARAMS("string='No String Supplied'")

    //Default parse command
    ON_PARSE_COMMAND(Default,CDispStrExtension,ITS_EMPTY)
    DEFAULT_PARSE_COMMAND(Default,CDispStrExtension)
END_PARSE_MAP(CDispStrExtension)
```

尽管只需要增加两行代码就能使这个例子工作，但我们真正关心的却是下面三行代码。ON_PARSE_COMMAND()宏允许定义一个新函数。注意，应提供函数名、所提供函数所在的类以及该函数使用的参数的类型。ON_PARSE_COMMAND()宏需要一个参数项，尽管你实际上可能并不需要任何参数就能使该函数正常运行。注意，因为缺省函数不需要任何参数，所以使用了 ITS_EMPTY 值。由于我们的新函数，DispStr——需要一个字符串指针，所

以它带有一个参数 ITS_PSTR。

然后下一行是 ON_PARSE_COMMAND_PARAMS()宏。你必须告诉 Visual C++如何处理你的函数的参数。例如，如果我们想强制用户为函数提供一个串值，就应在 ON_PARSE_COMMAND_PARAMS()宏中，简单的放上 String。因为实际上并不一定要求用户提供一个字符串才能使用函数工作，所以这里提供了一个缺省值 。注意如果你需要一个参数而用户没有提供的话，查询就会失败。最后还要用 DEFAULT_PARSE_COMMAND() 宏告诉 Visual C++用哪个函数作为缺省函数。由于这个例子的 Default 函数挺合适，所以就没有更改这个缺省的设置。

其次，你需要给 DispStr.H 文件增加一个函数项。除非通过修改类说明来包含你的新函数，否则 Visual C++对此将会一无所知，于是 DLL 也不会被编译。幸运的是，我们只要增加程序列表 13.2 中黑体显示的那一行代码就可以了。程序列表 13.2 列出了该行代码周围的代码行，以便于你知道新的一行应添加在什么地方。

程序列表 13.2

```
{
public:
        CDispStrExtension();
        ~CDispStrExtension();

//Overrides
        //ClassWizard generated virtual function overrides
```

```
        //NOTE - the ClassWizard will add and remove member functions here.
        //    DO NOT EDIT what you see in these blocks of generated code !
    //{{AFX_VIRTUAL(CDispStrExtension)
    public:
    virtual BOOL GetExtensionVersion(HSE_VERSION_INFO* pVer);
    //}}AFX_VIRTUAL
    //TODO: Add handlers for your commands here.
    // For example:

    void Default(CHttpServerContext* pCtxt);
    void DisplayStr(CHttpServerContext* ptext,LPTSTR pszString);

    DECLARE_PARSE_MAP()
    //{{AFX_MSG(CDispStrExtension)
    //}}AFX_MSG
}
```

可以看出，增加函数调用声明很简单。不过现在你可能还不清楚声明的 CHttpServerContext* pCtxt **部分是从哪里来的。我们确实没有在前面的任何分析映射宏中对它进行说明。实际上** pCtxt **参数是缺省地传递给函数的。记住前面已经介绍过** CHttpServer **类可以为每个用户请求创建一个** CHttpServerContext **对象，这就是该参数的来源。你得到的是指向与用户对函数的调用相联的** CHttpServerContext **对象的指针。这也是对函数的多个调用进行区分的办法——**

每个调用都有一个完全不同的对象与之相关联。

制作这个 DLL 函数还剩最后一件事——给 DispStr.CPP 文件添加函数代码。这个例子是将函数代码放在已有的 Default()函数代码之后。程序列表 13.3 显示的是这个例子使用的非常短的函数。这里没有什么花里胡哨的东西，其全部目的在于说明如何完成这件任务。

程序列表 13.3

```
void CDispStrExtension::DisplayStr(CHttpServerContext *pCtxt,LPTSTR pszString)
{
    // Start sending information to the new Web including a defaullt title.
    StartContent(pCtxt);
    WriteTitle(pCtxt);

    //Display the body of the new page.
    * pCtxt << _T("<H3><CENTER>ISAPI Server Extension Example</CENTER></H3><P><P>");
    * pCtxt << _T("This was the string you entered:<P><EM>);
    * pCtxt << _T(pszString);
    * pCtxt << _T(pszString);
    * pCtxt << _T("</EM>");

    //End the display area.
```

```
    EndContent(pCtxt);
}
```

这个函数本身很容易理解。我们做的第一件事是告诉 Visual C++启动一个 Web 页，这就像给文档加入<HTML>和<HEAD> 标记一样。我们做的第二件事是输出一个标题，就像给文档加入<TITLE>标记一样。重载缺省标题的唯一途径是重载 WriteTitle()函数，这是一件想干就能干成的事。现在有了标题，就该创建内容了。需要使用流操作符来发送信息。注意这个例子很轻松自如地使用了第 8 章介绍的所有标记。对标准 HTML 文档能做的事也同样能够对 ISAPI 扩展做。待会儿将会说明这些标记是如何一起工作从而产生了 Web 页的。另外还应注意我们将从 Web 页得到的字符串发送回新 Web 页，甚至不必将值转换为文本，就可以使它们工作了。这个例子中使用的最后一个函数调用是 EndContent()，它告诉 Visual C++已经完成发送信息的任务了，这就象在文档末尾加上</HTML>一样。

现在你就可以编译我们创建的 ISAPI 扩展了。一旦编译成功，就可以将 DLL 移到 Web 服务器上。有好几个逻辑位置可以存放该 DLL，不过最常用的两个位置是 Scripts 目录或一个特殊的 Controls 目录。我通常将自己的所有控件都放在 Controls 目录中以便于查找，不过实际的存放位置并不重要。唯一的原则是用户应该能够通过 Web 站点来访问包含该 DLL 的目录，而且你应该用 Web 服务器提供的 Internet 服务管理器将该目录标记为可执行的。

13.3 在 Web 页中使用 ISAPI 扩展

现在 Web 服务器上已经有了一个新的 ISAPI 扩展，可以进行测试了。记住该 ISA 有两个函数：由 ISAPI Extension Wizard（ISAPI 扩展向导）提供的缺省函数(Default)和我们加入的函数(DisplayStr)。另外，DisplayStr()函数还将依据用户是否为函数的工作提供数据来决定采取两种不同的工作方式中的哪一种方式。程序列表 13.4 显示了创建这个例子的 ISA 的测试 Web 页所需的 HTML 代码。存放该代码的文件采用.HTML 作为扩展名。

程序列表 13.4

```
<HTML>
<HEAD>
<TITLE>ISAPI Extension Example</TITLE>
</HEAD>
<!- 创建显示我们的按钮的窗体 ->
<FORM NAME="MyForm">
<!- 显示一个标题 ->
<H3><CENTER>Display a String ISAPI Extension</CENTER></H3>

<!- 显示一个编辑框 ->
Type a string value:
```

```
<INPUT TYPE=TEXTBOX NAME="StringValue" SIZE=40><P>

<!- 显示一些按钮 ->
<CENTER>

<!- 执行缺省行动 ->
<INPUT LANGUAGE="JavaScript"
        TYPE=BUTTON
        VALUE="Default"
        ONCLICK="window.location.href = 'controls/dispStr.dll?'">
<!- 用一个串执行缺省行动 ->
<INPUT LANGUAGE="JavaScript"
        TYPE=BUTTON
        VALUE="Display String"
        ONCLICK="window.location.href = 'controls/dispStr.dll?DisplayStr?'
                +MyForm.StringValue.value">
</CENTER>
</FORM>
</HTML>
```

可移植性 尽管 Internet Explorer 能很好地显示在 HTLM 文档主体中<BODY>和</BODY>标记之间的<INPUT>标记，但 Netscape Navigator 甚至都不认识它们。如果想做一个适用于所有浏览器的格式，就必须将<INPUT>标记放在<FORM>和</FORM>标记对之间的窗体中。

样本代码并不复杂。它包括一个标题、一个编辑框和两个按钮，其中编辑框将包含用户想显示的串的值。注意，我们的格式中并没有包括提交动作，而是利用标准的按钮和 Java Script 来启动该动作。在使用像这个例子一样包含多个函数的 ISA 时，这样安排比提交似乎要更好一些，我们就是这样做的。

另外，对每个按钮的 ONCLICK 事件所提供的 JavaScript 还要注意以下两点。对 Default（缺省）按钮只是调用 DispStr.DLL 本身，而没有指定一个特殊的函数。另一方面，Display String（显示字符串）按钮却同时包含一个函数名和一个串参数作为 DispStr.DLL 调用的一部分。待会儿将简单地交待一下它是如何工作的。

技巧 HTML 并不把编辑框中空格之后的文本发送给服务器。例如，如果在编辑框中输入 Hello World，服务器只会接收到 Hello。必须在单词间放上加号(＋)才能将它们发给服务器。例如，如果给服务器发送 Hello+World，服务器看到的就是 Hello World（Web 服务器日志仍会显示+符号，它表示是在到达服务器后加号才被空格替换的）。这意味着在向服务器发送串之前，用在例子程序中添加的一些不被显示的文本操作符代码（如加号）去替换编辑框中的空格。

打开测试 Web 页，看到的是如图 13.1 显示的页面。如前所述，这是一个测试页，所以为简单起见，并没有添加太多的格式方面的代码。

注释 一定要注意，把服务器上用来保存 DispStr.DLL 的目录，设置成 Web 服务器在因特网服务管理器内可执行的特权，否则，如果浏览器试图执行该文件时就会出现许多错误——而大部分都不告诉你是由于它不能执行该文件（最常见的错误是说该服务未被支持）而引起的。将目录置为在操作系统层次可执行的特权还不足以允许 Web 站点的访问者使用你创建的 ISA。大部分情况下，将 ISA DLL 放在 Web 服务器的脚本目录（假定有一个）下，就能避免出现以上问题。

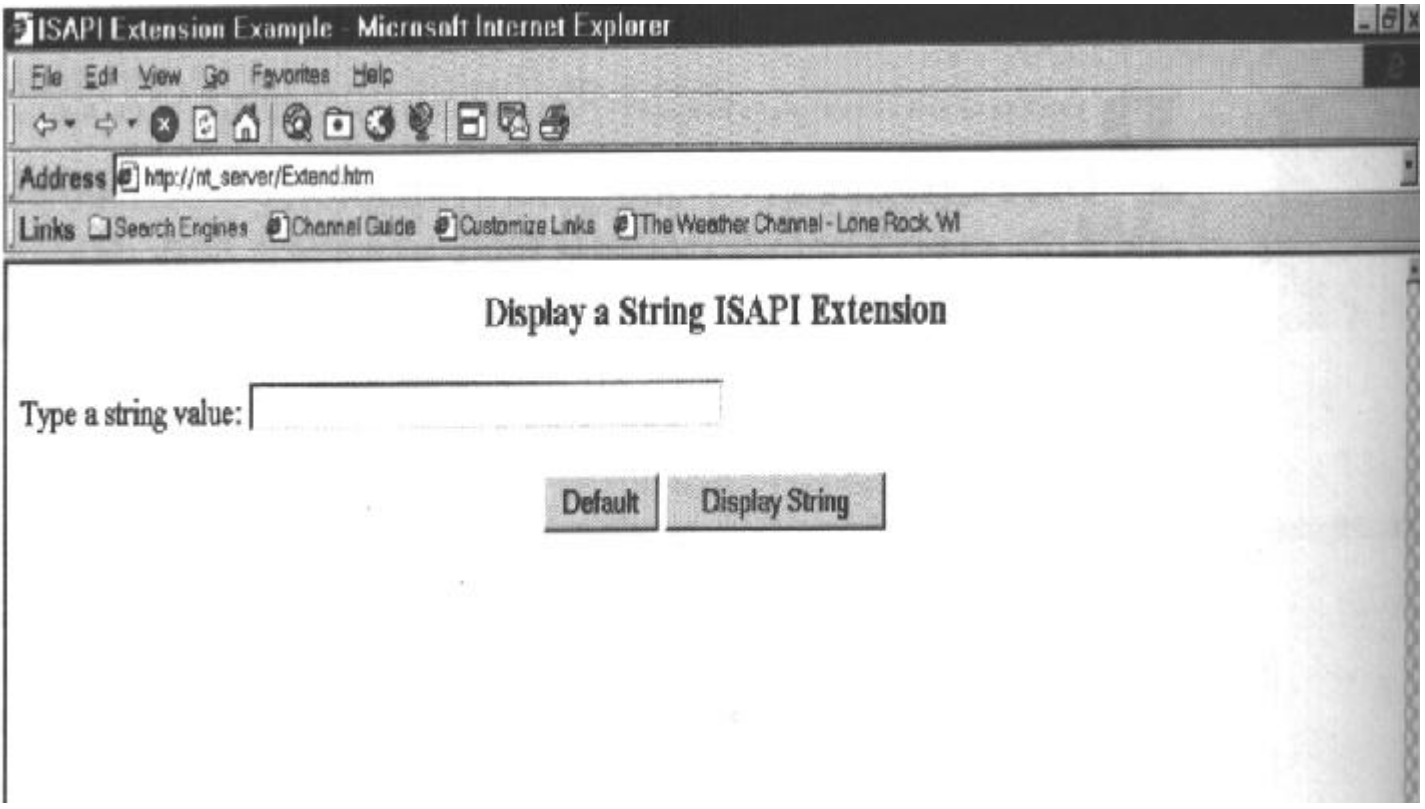
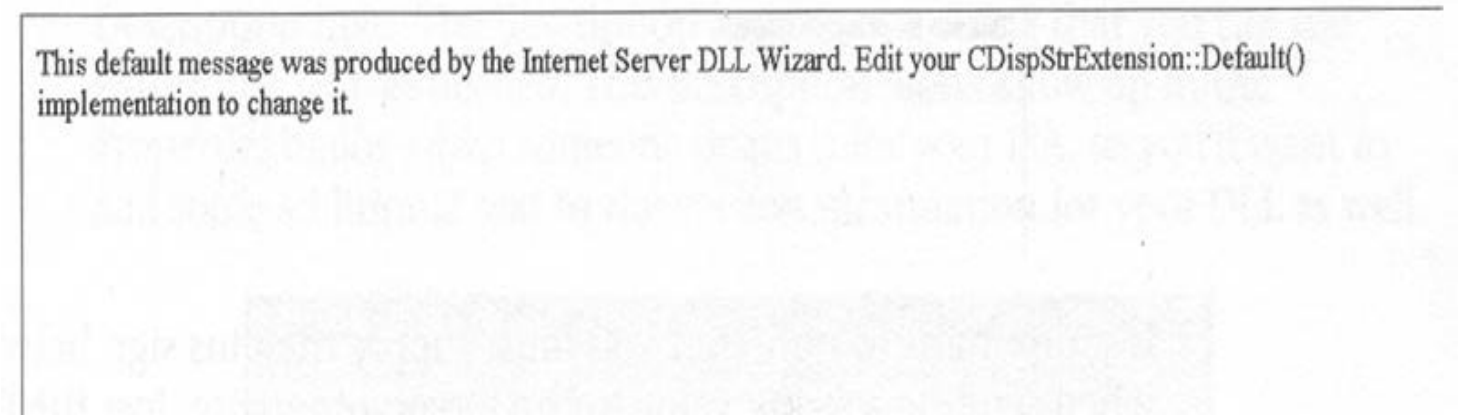
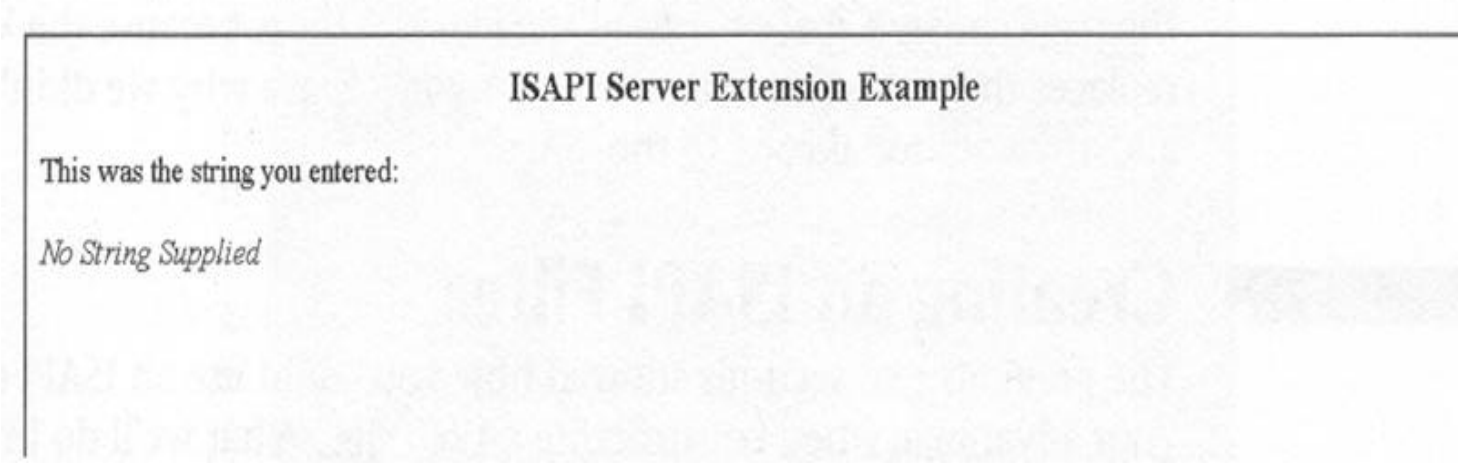


图 13.1　**这个简单的测试页面允许我们检查** DispStr.DLL **的所有功能**

现在开始测试 DispStr.DLL**。首先单击** Default **按钮，就会看到如下图所示的由** ISAPI Extension Wizard（ISAPI **扩展向导）提供的缺省字符串（实际上，它通知你必须给缺省函数增加一些代码）。**



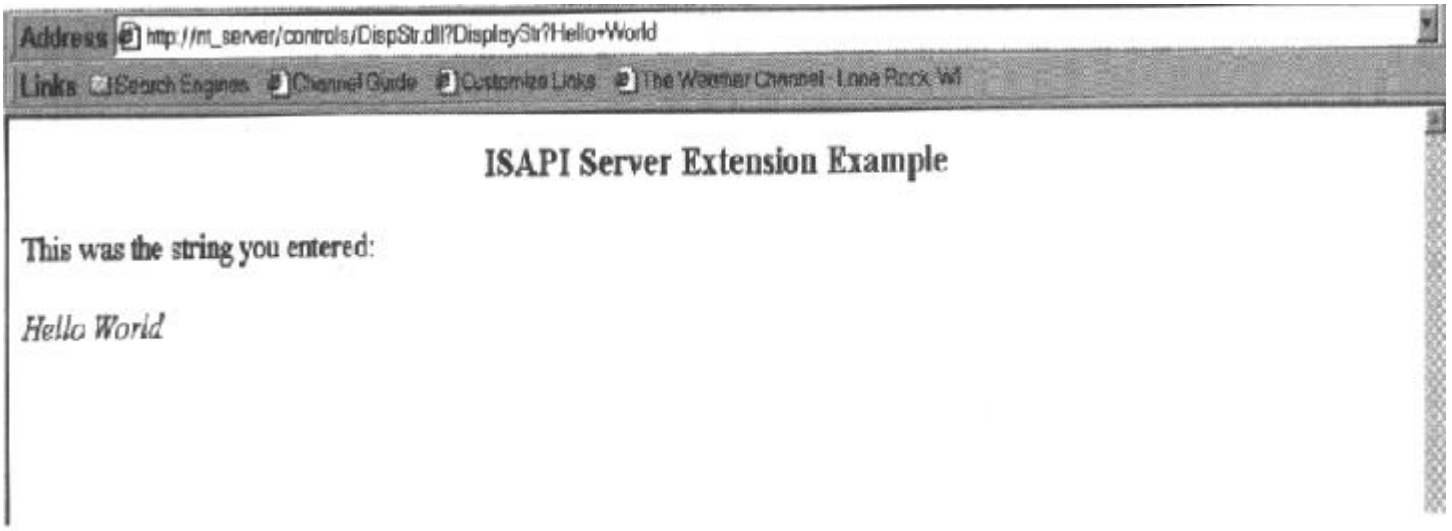
单击 Back **按钮（或浏览器上与之功能相同的部件），就回到** Extend.HTM **页。现在单击** Display String **按钮，就能看到如下图所示的结果。**



这个例子用 DisplayStr **函数调用** DispStr.DLL**，但是因为编辑框中没有任何值，所以函数必须显示前面作为** ON_PARSE_COMMAND_PARAMS()**宏的一部分而定义的缺省字符串。现在你应该明白大多数情况下提供一个缺省字符串的重要性了。如果用户忘记了提供一个需要的值，也不必担心会使** ISA **产生问题。相反它们得到的将是一个可以帮助解决** Web **页问题的返回值。这样用户就会确切知道哪里错了，从而只需花几分钟就能解决问题，而不是像通常那样花几个**

小时去进行调试。

再次单击 Back 按钮回到 Extend.HTM 显示页面，这次在 Type a String Value（键入一个串值）域中输入 Hello+World。单击 Display String （显示串）按钮就能看到如下图所示的结果。这里我们使用 DispStr.DLL 的 DisplayStr 函数并提供 Hello+World 作为字符串值。



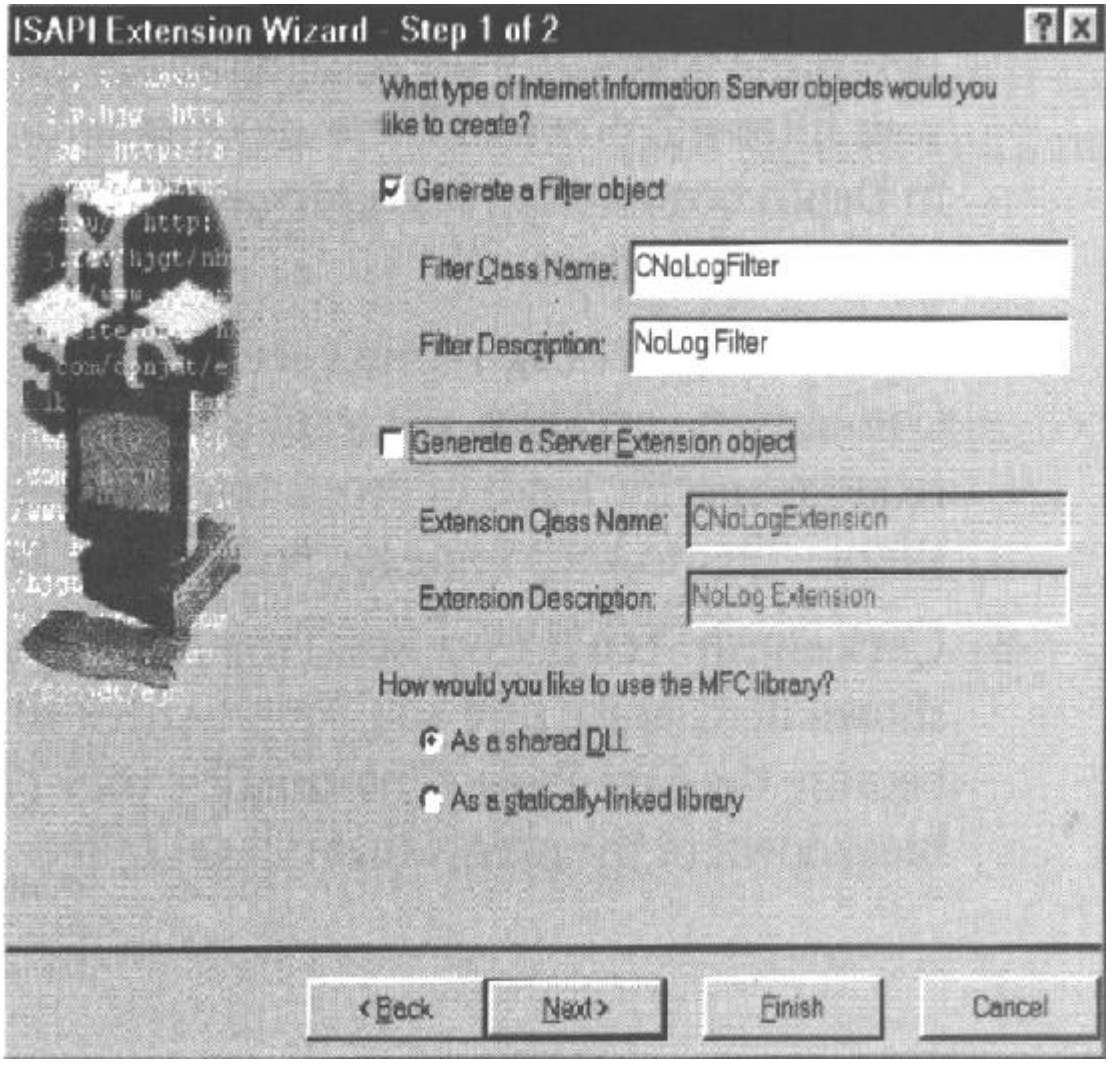
在向服务器发送字符串值时，你必须在单词间加上加号，否则将只显示第一个单词（你可以亲自做个实验。尝试不同的组合时，一定要检查 Web 服务器的日志）。如果看一下浏览器上的 Location 域，就会注意单词间有加号，但页内容中单词间却是空格。这是因为 Web 服务器用空格替换了加号，这也正是不必给 ISA 再添什么代码的原因。

13.4 创建 ISAPI 过滤器

前面两节介绍了如何利用 ISAPI 扩展更好地构造 Web 站点。本节将介绍构建自己的 ISAPI 过滤器的过程。创建 ISAPI 过滤器过程与创建扩展的过程的起始部分是一样的。下面的步骤将帮助你创建一个程序框架。

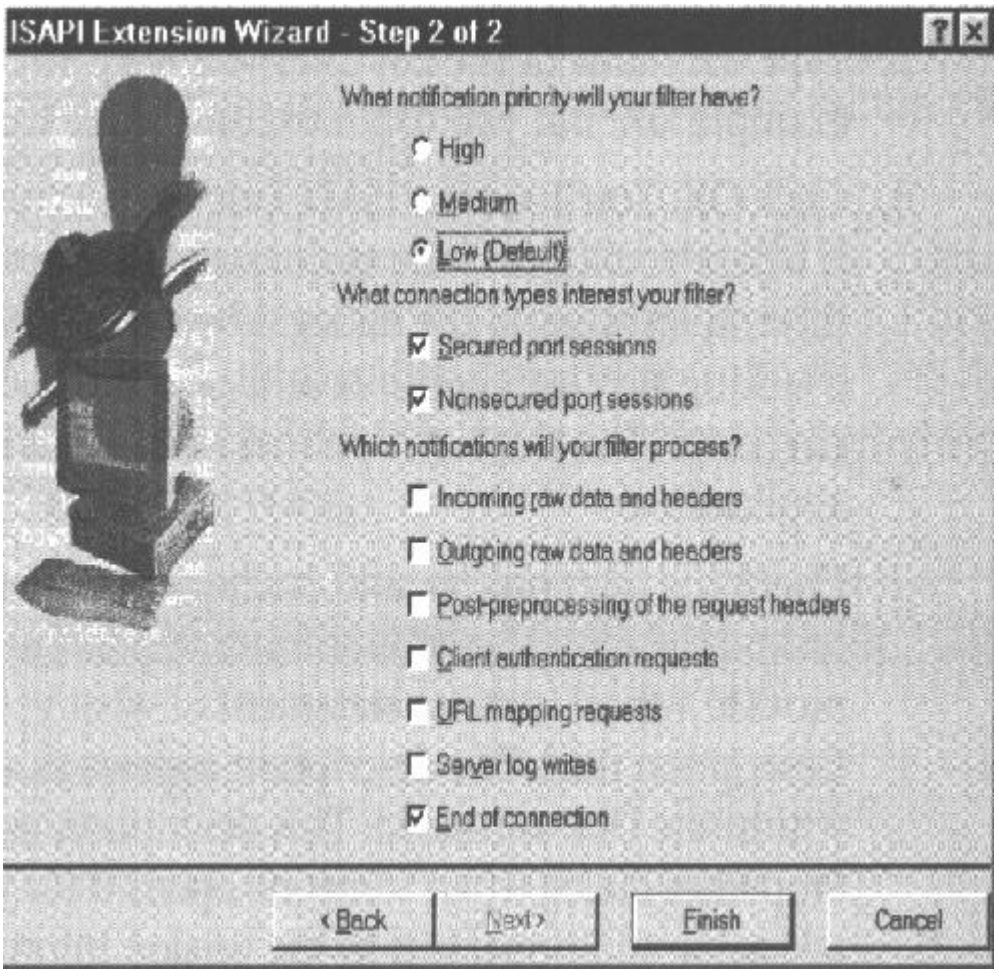
注 记住 ISAPI 过滤器与扩展不同，过滤器在特定事件发生时自动做出反应，而扩展更像是一个应用程序那样被调用。

1. 使用 File（文件）|New（新建）命令显示 New 对话框。选中 ISAPI Extension Wizard（ISAPI 扩展向导）（即使要创建的是过滤器，也要用 ISAPI Extension Wizard 来创建）。
2. 输入想创建的过滤器的名称。对这个 ISA 示例我们使用了 NoLog，但你可以任意选用其它名称。
3. 单击 OK，即出现 ISAPI Extension Wizard（ISAPI 扩展向导)——Step 1 of 2 对话框。在这里为你的 ISA 选择各种特性。注意对话框有三个主要区域。选中第一个复选框表示创建过滤器，选中第二个复选框表示创建扩展。第三部分定义如何将 MFC 链接到你的应用程序中。
4. 选中 Filter（过滤器）选项而不选中 Extension （扩展）选项（注意加入过滤器也就增加了一些其它步骤）。在 Filter Description（过滤器描述）域中，提供一个描述你的 ISA 功能的短小扼要的说明。该描述将作为字符串来显示，需要时可以在 DLL 中使用它。当别人为了使用 ISA 打开它时，该描述不会出现在 Properties（属性）对话框中，所以还要给 DLL 的版本信息中加入一些文本。



5. **在** Filter Description **(过滤器描述)域中输入** Classify some log entries for security reasons **（为了安全原因而对日志项进行密级划分）。**

6. **单击** Next**，就能看到如下图所示的** ISAPI Extension Wizard **（**ISAPI **扩展向导）**——Step 2 of 2 **对话框。**

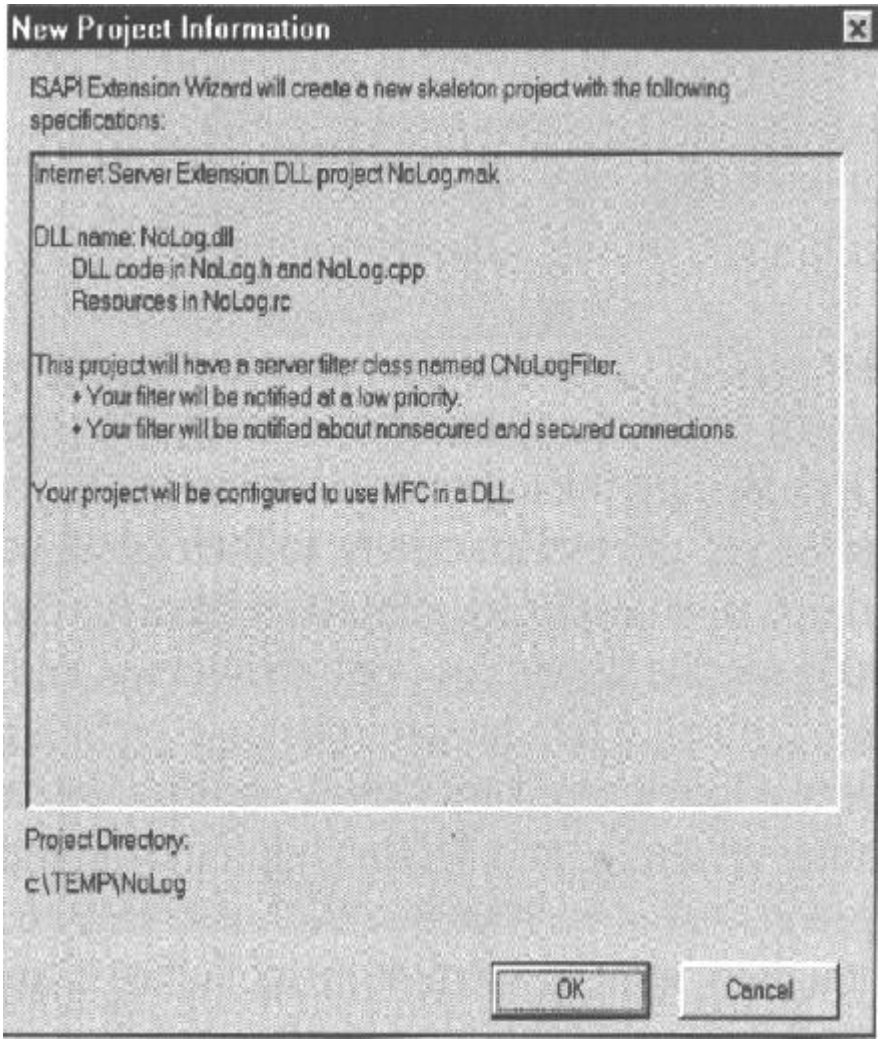


这是选择过滤器能够提供的事件以及监测类型的属性页。有三个方面的情况需要考虑。这些选项的详细介绍请参见“高级技巧”中的“选择过滤器选项”部分。

7. **选中** Server log Writes（**服务器日志写入**）**复选框，而不选** End of Connection（**连接结束**）**复选框。由于我们创建了一个简单的过滤器来保持某些日志项的密级划分，所以每当用户申请服务器的特殊类型服务时就激活过滤器。**

8. **单击** Finish，**就会看到如下图所示的** New Project Information （**新建工程信**

息）对话框。一定要二次核对过滤器的这些设置项，因为 New Project Information 页提供了对过滤器将要看到的事件进行的详细分类。



9. 单击 OK，ISAPI Extension Wizard（ISAPI 扩展向导)就会创建所需的程序框架。

这个示例的代码部分简单明了。过滤程序很容易弄得复杂化，从而导致调试很困难。大多数情况下，你应该让过滤器程序尽可能简短。在对出错的过滤

器查错时，如果程序是模块化的，将会提供极大的帮助。程序列表 13.5 列出了为使 OnLog() 函数工作而需要添加的代码。

程序列表 13.5

```
DWORD CNoLogFilter::OnLog(CHttpFilterContext *pCtxt,PHTTP_FILTER_LOG pLog)
{
    //Check for a specific notification.
    if (strstr(pLog->pszTarget,"NoLog.htm") !=NULL)
    {
        //Change the target information to Classified.
        pLog->pszTarget = "Classified Target";
        pLog->pszParameters = "Classified Parms";
    }

    //Pass control to the next filter.
    return SF_STATUS_REQ_NEXT_NOTIFICATION;
}
```

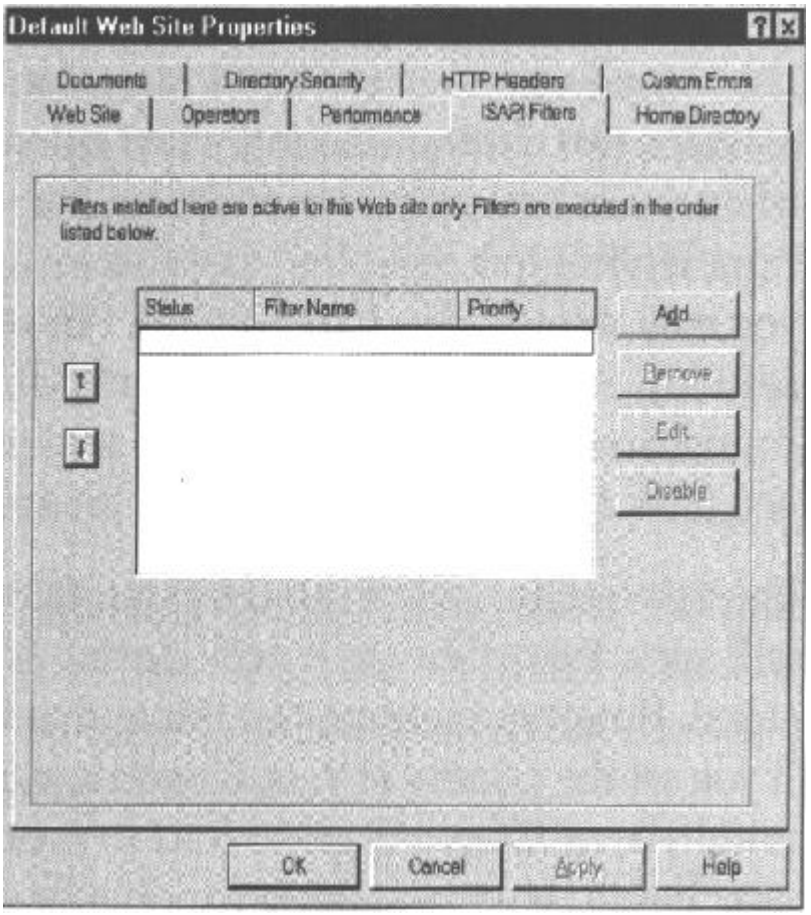
可以看出，为了某个特定目标，我们对日志项进行简单的监测，这个特定目标可以是除另一个 Web 页之外的任意数量的事情。一旦找到所需的日志项，我们将按照分类设置修改两个成员：pszTarget 和 pszParameters。显然这种思路已大大简化，不过都有实际的用途。你可能不想让某个特定日志项被别人偷看，

这里介绍的方法就可以实现你的目的。记住黑客们经常利用它们在服务器上找到的目志文件，来盗窃公司的内部资源。不仅如此，对某些日志项进行密级划分还有其它原因。在某些情况下，你可能想完全删除某些日志项只是因为不想再监测它们。好了，我们已经详细说明了如何小心使用这种过滤器，不过这将是一个好的开端。

现在可以编译新的ISAPI 过滤器了，但是，在你将它移到服务器上的 Controls 或 Scripts 目录之后，仍要做一件事。与扩展不同，ISA 过滤器在你启动服务器时即被装载。也就是说，如果你使用的是老版本的 IIS，那么，就停止目标服务器，生成一个注册表项，然后再次启动该服务器。过滤器将作为启动进程的一部分被装载。

WWW 服务在如下的注册值中保存它的过滤器项：HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\W3SVC\Parameters\Filter DLLs。你可能已经在该位置找到一个或多个值。这里要做的是加上一个逗号，然后输入自己新的 ISA 过滤器的位置。一定要在学习下节之前做这一步，否则过滤器就不会被装载。实际上，如果过滤器工作出了问题，这里是你首先应该检查的地方。

对 IIS 4.0 以上版本的用户来说，不必为装载 ISAPI 过滤器而手工编辑注册表了。此时所要完成的工作就是打开 Microsoft Management Console（管理控制台），用右键单击需添加过滤器的 Web 站点，从关联菜单中选择 Properties（属性），再选中 ISAPZ Filter（过滤器）项，就会看到如下所示的对话框。



单击 Add（**添加**）**按钮，就会看到** Filter Properties （**过滤器属性**）**对话框，在** Filter Name （**过滤器名**）**域中键入名称，使用** Browse（**浏览**）**按钮找到** Nolog.DLL **文件（或想添加到** Web **站点的随便什么别的** ISAPI **过滤器）。选定文件后，其名称将出现于** Executable（**可执行**）**域。单击** OK **关闭** Filter Properties （**过滤器属性**）**对话框；再次单击** OK **关闭** Default Web Site Properties（**缺省** Web **站点属性**）**对话框。**IIS **将为你自动加载该过滤器。**

高级技巧

选择过滤器选项

ISAPI Extension Wizard (ISAPI 扩展向导)——Step 2 of 2 页对于过滤器设计者是非常重要的，这是因为它包含了需要为过滤器设置的监测选项。这一页上的选项分三部分。第一部分确定过滤器优先级，缺省设置为 Low （低），这在大多数情况下运行良好，一般后台任务使用这一层次的优先级。安全过滤器需要设置为 medium（中）优先级，毕竟你不想让过滤器在事件之后再作出反应，在事件发生期间过滤器就作出反应就好得多了。最后，High（高）优先级设置应为关键层次的过滤器保留着，例如，向每个人发出一条消息，声明由于电源故障，服务器即将关机，这时就应使用 High 优先级了。

ISAPI Extension Wizard (ISAPI 扩展向导)—Step 2 of 2 对话框的第二部分包含着两个选项。如果用户拥有安全的连接就应复选第一项，例如，如果计划对注册用户提供额外服务，而不让匿名用户得到这些额外服务，那么就应复选第一项了。第二项是不安全端口会话，允许你在用户不拥有对服务器的安全连接时监测事件。如果你有一个 Internet 一般 Web 站点，大多数用户都是这样去访问你的站点的。

第三部分是过滤器追踪的事件列表，每当这里指定的事件发生，就调用过滤器，但是，过滤器被调用时有两件事要影响它。如果过滤器的优先级为 Low（低），那么 High（高）或 medium（中）优先级的过滤器将先对事件作出反应。另外，你的过滤器不得不设置成用户当前安全级来监测事件，换言之，如果你把过滤器设置成监测非安全级行动而用户处于安全模式，则过滤器就不会被调用。

13.5 使用 ISAPI 过滤器划分 Web 站点的密级

现在 ISA 过滤器已经完成并且编译好了，应将它放在自己通常存放过滤器的位置，可以是 Controls、Scripts、甚至 System32 目录下。记住要在服务器停止运行时生成所需的注册表项，完成后再重新启动服务器。现在我们开始测试过滤器。

我们首先需要完成的工作是创建三个 Web 页。第一个是中心 Web 页，我们使用它在另两个页（一个正常注册，另一个按密级注册）之间切换。第一个 Web 页称为 Filter.HTM。程序列表 13.6 显示了该测试部分的源代码。

程序列表 13.6

```
<HTML>
<HEAD>
<TITLE>ISAPI Filter Test</TITLE>
</HEAD>
<BODY>
```

```
<!- 显示标题 ->
<H3><CENTER>ISAPI(NoLog)Filter Tester</CENTER></H3>
<!- 显示链接和某些解释性文本 ->
Choosing the first link will create a classified log entry,
while the second one creates a standard log entry.<P>

Log this Link as <A HREF="NoLog.htm">Classified</A><P>
Log this Link <A HREF="LogIt.htm">Normally</A>

</BODY>
</HTML>
```

这些代码并没有什么神秘可言，我们前面都曾使用过。该页包含一个标题，一些解释性文本以及几个链接。我们将使用它在两个测试页之间切换而无需太多额外的工作。

两个测试页的名字很重要，第一个测试页必须命名为 NoLog.HTM**，否则过滤器将不会工作（因为过滤器通过该名称来寻找目标，因此你必须提供一个名称）。程序列表** 13.7 **包含该页的代码。第二个测试页名为** LogIt.HTM**。程序列表** 13.8 **包含了创建它所需的源代码。两个测试页都包含一个标题和一些文本，但也仅此而已。**

程序列表 13.7

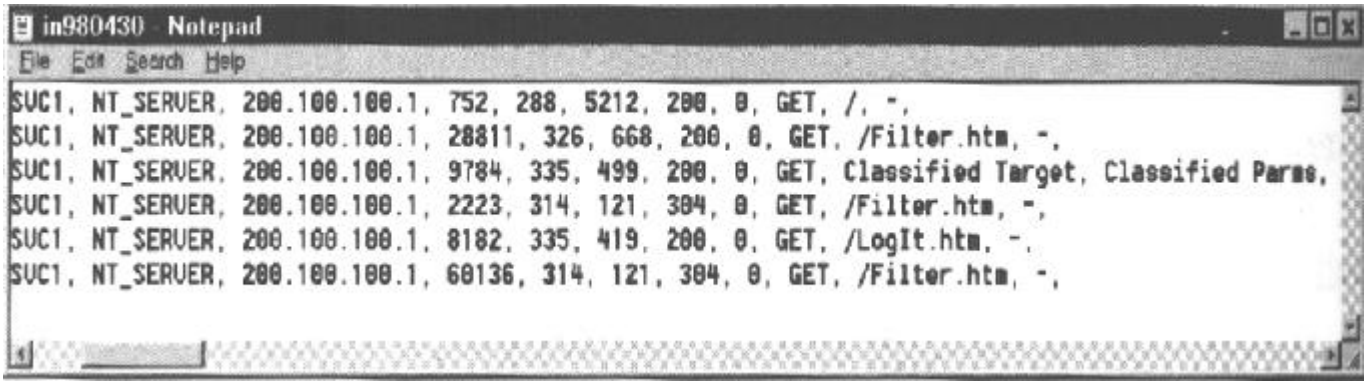
```
<HTML>
<HEAD>
<TITLE>ISAPI Filter Test</TITLE>
</HEAD>
<BODY>
<H3><CENTER>This Page is <EM>NOT</EM> Logged</CENTER></H3>
Which means that you won't see any of the normal entries in the log.<P>
What you will see are a bunch of classified entries instead.
</BODY>
</HTML>
```

程序列表 13.8

```
<HTML>
<HEAD>
<TITLE>ISAPI Filter Test</TITLE>
</HEAD>
<BODY>
<H3><CENTER>This Page is Logged</CENTER></H3>
Which means that you will see all of the normal entries in the log.
</BODY>
```

</HTML>

现在已经作好准备对 ISA 过滤器进行核查，你要做的就是用自己的浏览器打开 Filter.HTM web 页。一定要通过内部网连接访问 Web 服务器。在 Explorer 中双击 Filter.HTM 并将它作为文件打开，这样就不会迫使 Web 服务器做出日志项。在两个测试页之间切换几次后可以打开日志文件。如果你的 Web 服务器设置和我的一样，那么你的 Web 服务器的 System32\LogFiles 目录中每天都应该有一个日志文件。下图就是测试 NoLog ISA 过滤器时的日志项。



请注意，除了 NoLog.HTM 的日志项之外，所有日志项都显示了正确的 Web 页。图中唯一的 NoLog.HTM 项位于第三行（从上往下看），由于 NoLog 过滤器已经修改了日志项，因此，这些项均按密级划分进行标记。另外，LogIt 和 Filter 项都能正常显示。显然这些都是自动完成的，过滤器在后台对日志事件自动作出反应。

13.6 使用 ISAPI 扩展转发服务器信息

几乎每个人都知道本地日期和时间，看一下时钟就够了。如果你是网络管理员或 Web 站点拥有者，管理远程资源时，只知道本地时间通常就不够用了。实际上，更重要的是要知道所辖位置的日期和时间。

下面的示例说明了如何从服务器上取得日期和时间并在浏览器窗口中显示它们。使用这个特定 ISAPI 的方式有多种，但我们讨论的是在一个简单 Web 页上使用它们。你顺便可以获得的最重要概念是，使用 ISAPI 扩展，允许你访问那些你坐在控制台前使用正规程序访问的每一条信息。换言之，坐在计算机前能做的一切也可以远程地管理它们。实际上唯一的限制是你应考虑到标准 Windows NT 的安全性。

技巧 灵活性是一把双刃剑。对于 ISAPI 扩展要考虑一些对普通程序不必考虑的限制。例如，除非你拥有实际管理整个进程的全部能力，否则就要对是否允许远程备份进行再三考虑。坐在 Windows NT 控制台前的管理员能够通过物理地验证磁带编号或其他识别信息看到，你打算使用的磁带不是你用于备份应该使用的那盘磁带，问题是，你不得不问问自己，远程管理员是否也具有这种能力，多数情况下，答案都是否定的。

创建得到时间/日期的 ISAPI 扩展

让我们设计基本程序吧。由于本章中已设计过一个基本的 ISAPI 扩展，本节中不再花费时间讨论程序的细节。请你按下列步骤进行操作吧。

1. 创建一个新的 ISAPI 扩展工程，我把它命名为 GetTD，至于你怎么命名就随你的便了。
2. 对于 ISAPI Extension Wizard（ISAPI 扩展向导）Step 1 of 1 没有什么特别可做的事，单击 Finish（完成）即可。
3. 单击 OK，关闭 New Project Information（新建工程信息）对话框，在这之前应先确认所有程序参数。

下面创建代码。和其他 ISAPI 扩展一样，这个例子也要添加三段主要代码。下面的列表就是要添加的东西。

- Parse map（分析映射） 分析映射对 DLL 中的函数如何调用作出定义，它还定义了所需变量以及这些变量映射到你的函数中的方式。
- Function（函数） 显然，如果没有某些向浏览器提供输出的代码，你的 DLL 就没用了。
- Function definition（函数定义） 需要在 ISAPI 扩展的头文件中添加函数说明。

正如你要看到的，我们将对 GetTD.CPP 文件作两处改动，还要对 GetTD.H 文件作一处改动。程序列表 13.9 是 GetTD.CPP 文件的有关代码。对分析映射及 GetTD()函数都要看一下。还要注意对 Default()函数所作的改动，它使得当你在一个 ISAPI 扩展文件中有多个函数时这个函数更有用。程序列表 13.10 列出了

整个 GetTD.H **文件。**

程序列表 13.9

```
//////////////////////////////////////////////////////////////////////
//command-parsing map

BEGIN_PARSE_MAP(CGetTDExtension,CHttpServer)
        // Get the system time and date.
        ON_PARSE_COMMAND(GetTD,CGetTDExtension,ITS_EMPTY)

        // Default parse map.
        ON_PARSE_COMMAND(Default,CGetTDExtension,ITS_EMPTY)
        DEFAULT_PARSE_COMMAND(Default,CGetTDExtension)
END_PARSE_MAP(CGetTDExtension)

//////////////////////////////////////////////////////////////////////
// The one and only CGetTDExtension object

CGetTDExtension theExtension;

//////////////////////////////////////////////////////////////////////
// CGetTDExtension implemenation
```

```
CGetTDExtension::CGetTDExtension();
{
}

CGetTDExtension::~CGetTDExtension();
{
}

BOOL CGetTDExtension::CGetTDExtensionVersion(HSE_VERSION_INFO* pVer)
{
        // Call default implementation for initialization
        CHttpServer::GetExtensionVersion(pVer);

        // Load description string
        TCHAR sz[HSE_MAX_EXT_DLL_NAME_LEN+1];
        ISAPIVERIFY(::LoadString(AfxGetResourceHandle(),IDS_SERVER, sz,
                                HSE_MAX_EXT_DLL_NAME_LEN));
        _tcscpy(pVer->lpszExtensionDesc,sz);
        return TRUE;
}
```

```
////////////////////////////////////////////////////////////////////
// CGetTDExtension command handlers

void CGetTDExtension::Default(CHttpServerContext* pCtxt)
{
        StartContent(pCtxt);
        WriteTitle(pCtxt);

        // display an informational message on how to use this extension.
        * pCtxt << _T("<H2><CENTER>How to Use This ISAPI Extendion </CENTER>
</H2>");
        * pCtxt << _T("Use the <STRONG><EM>GetTD</EM></STRONG>function to");
        * pCtxt << _T("retrieve the current Web server date and time.<P>");

        EndContent(pCtxt);
}

void CGetTDExtension::GetTD(CHttpServerContext* pCtxt)
{
        // Creat a couple of local variables.
```

```
CString  cDate;                    // String representation of date.
CString  cTime;                    // String representation of time.
CTime  oCurDate;                   //  CUrrent  date  and time.

// Get the current date and store it in a string.
oCurDate = CTime::GetCurrentTime();
cDate = oCurDate.format("%A,%d %B %Y");
cTime = oCurDate.format("%I:%M %p");

// Start an HTML page.
StartContent(pCtxt);
WriteTitle(pCtxt);

// Display the time and date.
* pCtxt << _T("The system date is: ");
* pCtxt << _T("cDate);
* pCtxt << _T("<P>");
* pCtxt << _T("The system time is: ");
* pCtxt << _T("cTime");
* pCtxt << _T("<P>");
```

```
        // End the HTML page.
        EndContent(pCtxt);
}
```

程序列表 13.10

```
if  !defined(AFX_GETTD_H_E8B71025_2785_11D1_8E36_444553540000__INCLUDED_
)
define AFX_GETTD_H_E8B71025_2785_11D1_8E36_444553540000__INCLUDED_

// GETTD.H - Header file for your Internet Server
//      An ISAPI Extension that retrieves the time    and date.

#include "resource.h"

class CGetTDExtension : public CHttpServer
{
public:
        CGetTDExtension()
        ~CGetTDExtension()

// Overrides
```

```
	// ClassWizard generated virtual function overrides
		//NOTE - ClassWizard will add and remove member functions here.
		// DO NOT EDIT what you see in these blocks of generated code !
	//{{AFX_VIRTUAL(CGetTDExtension)
	public:
	virtual BOOL GetExtensionVersion(HSE_VERSION_INFO* pVer);
	//}}AFX_VIRTUAL

	// Default handler.
	void Default(CHttpServerContext* pCtxt);

	// Special time and date handlers.
	void GetTD(CHttpServerContext* pCtxt);

	DECLARE_PARSE_MAP()
	//{{AFX_MSG(CGetTDExtension)
	//}}AFX_MSG
};
//{{AFX_INSERT_LOCATION}}
//Microsoft Developer Studio will insert additional declarations immediately before the previous line.
```

```
#endif                                                    //!defined
(AFX_GETTD_H_E8B71025_2785_11D1_8E36_444553540000__INCLUDED)
```

让我们先讨论一下程序列表 13.9 中的代码。这个分析映射没有什么出人意料之处。我们不需要把来自客户机的数据传递给服务器，所以对 GetID 函数使用了 ITS_EMPTY 常量。

请注意我是如何修改 Default()函数的。通常如果 DLL 中只有一个函数时就要在代码中使用 Default()函数。但是，我发现，当 DLL 中多于一个函数时，作为生成 ISAPI 扩展接口文件的一种方法，使用 Default()函数也是相当方便的，这样，合作者(coworker) 能快速地构造一个 Web 页，查看 ISAPI 扩展提供的文档，然后去处理创建一个实际页的事务。

GetTD()函数是相当基础的一个函数。它只是获取系统时间并把它格式化成两个字符串，一个是时间，一个是日期。请注意我是如何使用 Format()函数来得到这两个字符串的。有了这两个字符串变量后，就可以创建包含其值的基本 Web 页面了。

程序列表 13.10 与我们以前的例子有很大区别。我做的一切工作就是说明 GetTD 函数。这是因为我没有向函数传递任何数据，所以唯一的数据是 pCtxt。

创建测试 GetTD 的 Web 页

现在看一看访问 ISAPI 扩展的 HTML 代码，程序列表 13.11 包含了一个非常简单的测试页面。

程序列表 13.11

```
<HTML>
<HEAD>
<TITLE>Get Time/Date ISAPI Extension Example</TITLE>
</HEAD>
<BODY>

<!- 显示标题 ->
<CENTER><H2>Getting Time and Date from Your Server</CENTER></H2>

<!- 显示到 Default()函数的链接 ->
Click <A HREF="Controls/GetTD.DLL?Default">Here</A> to see the Default( )
function message.<P>

<!- 显示到取得时间/日期页的链接 ->
Click <A HREF="Controls/GetTD.DLL?GetTD">Here</A> to see the Time( ) and
Date() function message.<P>

</BODY>
</HTML>
```

应该注意到程序列表 13.11 **比起以前的例子（程序列表** 13.4）**有一个主要的**

变化。这一次是使用了一个锚地去访问 ISAPI 扩展。用户单击这一链接后，得到的是由 ISAPI 生成的页，而不是像通常那样链接到 Web 服务器的一个静态页。这样做有几个重要的优点：

- 能够通过把检测到的浏览器类型作为参数传递给 ISAPI 扩展来创建浏览器专用页。大多数浏览器都提供了唯一识别信息。
- ISAPI 扩展允许定制 Web 页的内容来反映特定用户的需求。
- 使用 ISAPI 扩展，允许创建基于某些规则（如星期中某天）的链接。例如，能够把星期一欢迎页和星期五会见页连接到同一个主页链接上。

显然，使用类似于 ASP 或老版本的其他技术也能得类似的结果，但事实证明，在 Web 站点通常不使用的许多方式中，你也能使用锚地技术。

注释 记住先在内部网上测试这个 ISAPI 扩展。你不能在本地打开扩展，所以如果作为浏览器中的文件来简单地打开测试 Web 页的话，链接就会失败。

现在看一下这个 ISAPI 扩展的输出。图 13.2 显示了第一次看到的 Web 页。如你所见，它含有一个简单标题和两个链接，ISAPI 扩展中每个函数都有一个链接。

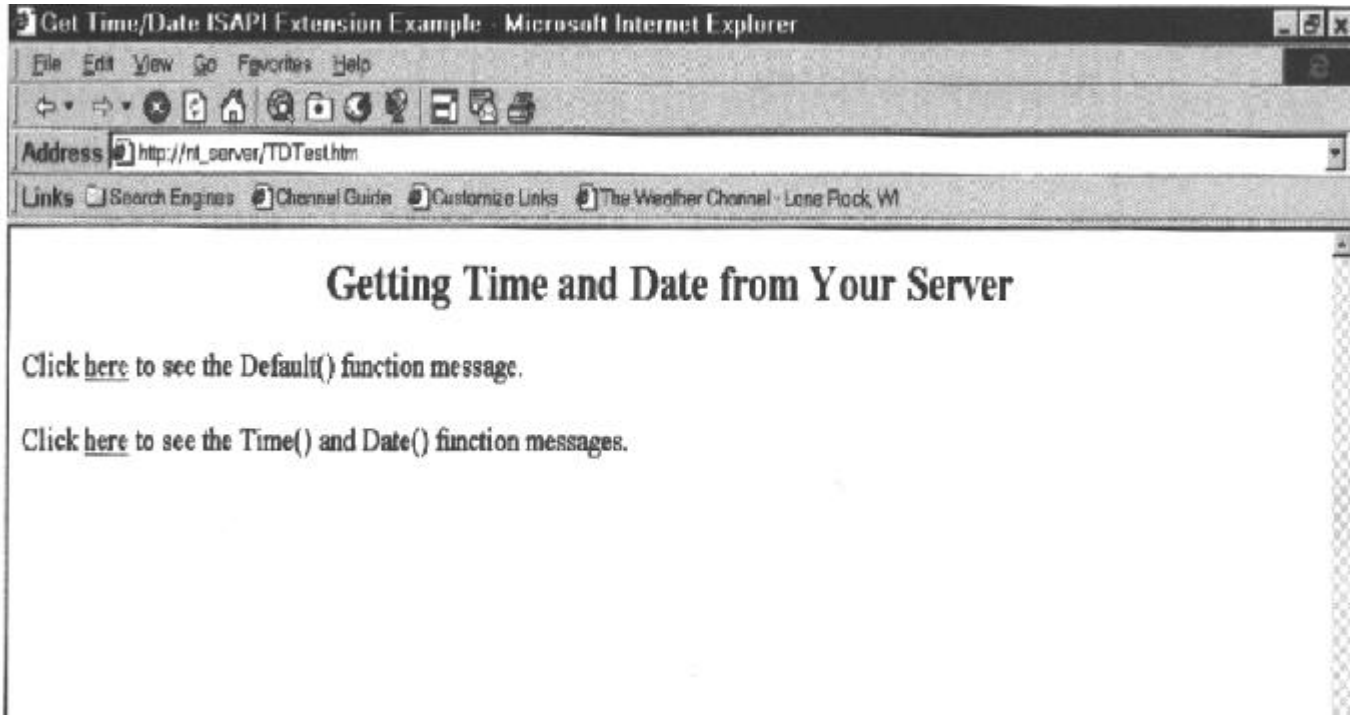
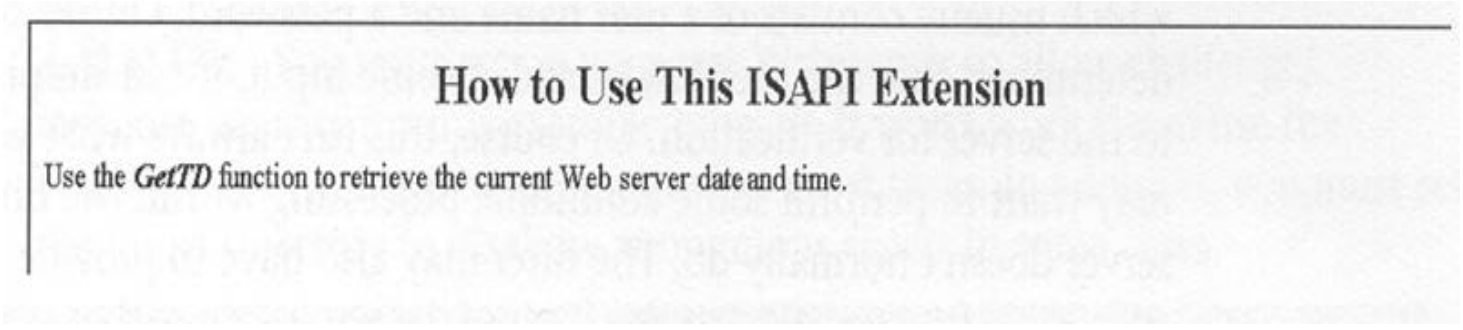


图 13.2　**初始的测试页有两个选项，一个去查看** Default()**函数的输出，另一个去查看** GetTD()**函数的输出**

如果单击第一个链接，你会看到如下图所示的显示内容：



尽管这一页没有给出 ISAPI **扩展的明显的使用指令，但你却知道了它能做些什么。显然，在产品型扩展中可以提供更多的细节，可以包括一个参数列表**

（如果有的话）作为 Default()函数页的一部分，有时，甚至可以包括一些与 ISAPI 扩展创建者进行联系的信息。

单击浏览字符串上的 Back（后退）按钮，退回到图 13.2 所示的初始 Web 页。现在试一试第二个链接。在试第二个链接之前，在测试服务器上变更一下时间和日期，这样做是为了证实 ISAPI 扩展确实是按所预期的那样完成了工作。下图是 GetTd()函数输出时间和日期的典型例子。如你所见，Web 页上显示的是服务器上变更了的系统时间，而不是本地时间。

```
The system date is : Saturday,02 May 1998
The system time is : 03: 13 PM
```

13.7 使用 ISAPI 过滤器请求用户名和口令字

存在许多不同的方式来使用 IIS 安全过滤器，但在完成其它工作前，让我们先了解一下如何制作一个简单的过滤器。本节帮助你理解如何创建一个过滤器，该过滤器在服务器上传递值，并使用这些值对用户的合法性进行证实。

过滤器需要做的第一件事是从用户那里得到输入，这些输入通常包含用户名和口令字。一旦过滤器确定用户已提供了某些输入，它就简单地把控制移交给服务器去进行验证工作。当然这并不总是你所想的结果。你也许想在过滤器内完成一些服务器通常不做的附加处理。过滤器也许还要提供某种形式的附加解密功能，或许还要有专门的错误消息处理功能。下列内容是安全过滤器能提供的附加特性。

- **解密** 我们总是选择在客户端对用户名和口令字进行加密，然后在服务器端对它们解密。ISAPI 过滤器恰好能够胜任这一工作。
- **认证** Windows NT 不强制使用内置的认证功能，你总是可以选择自己去认证用户并给予用户适当的安全级。于是，可以进行认证，然后把用户分到某一个组，之后让 Windows NT 负责细节。有许多不同的认证场合。
- **定制化错误处理** 也许你不喜欢 Windows NT 错误处理的方式，ISAPI 过滤器提供了限制最少的错误处理功能。当然，你提供的错误处理的精确层次由客户机和服务器来确定，这意味着你想对用户访问使用的浏览器实施控制。
- **专门化访问处理** 有时不想让每个人都能在 24 小时内随时访问 Web 站点。无论是 IIS 还是 Windows NT 目前都没有办法使你的服务器对于一般用户是关闭的而对另外的用户仍然是开放的。而 ISAPI 过滤器能基于当前时间或其它规则提供访问，甚至可以基于服务器加载来限制访问某些站点。

现在我们对过滤器能干些什么有了一个一般的了解。我们下面给出一个例子。首先是使用本章前一节中创建 ISAPI 过滤器的过程创建一个框架。创建工程时有一点差别，这个差别是该程序命名为 AuthUser，你可以随便起个其它什么名称。在 ISAPI Extension Wizard（ ISAPI 扩展向导）- Step2 of 2 对话框中，选中 Client Authentication Request（客户机认证请求）选项，并且不选中 End of Connection（连结终止）选项。程序列表 13.12 是要向 OnAuthenticate()函数添加

的代码。随后我们将对这些代码的作用进行说明。

注释 为了能够让本示例正常运行，必须把测试的 Web 服务器设置成允许 Chanllenge / Reponse 安全层次。如果你使用绝大多数情况下使用的 Windows NT Authentication 方法，那么该示例将不能正常工作。另外，还必须把目标目录设置成不允许匿名访问的方式。

程序列表 13.12

```
// AUTHUSER.CPP - Implementation file for your Internet Server
// AuthUser Filter

#include "stdafx.h"
#include "AuthUser.h"

////////////////////////////////////////////////////////////////////
// The one and only CWinApp object
// NOTE: You may remove this object if you alter your project to no
// longer use MFC in a DLL

CWinApp theApp;

////////////////////////////////////////////////////////////////////
```

```
// The one and only CAuthUserFilter object

CAuthUserFilter;

////////////////////////////////////////////////////////////////////
// CAuthUserFilter implementation

CAuthUserFilter::CAuthUserFilter()
{
}

CAuthUserFilter::~CAuthUserFilter()
{
}

BOOL CAuthUserFilter::CGetFilterVersion(PHTTP_FILTER_VERSION pVer)
{
	// Call default implementation for initialization
	CHttpFilter::GetFilterVersion(pVer);

	// Clear the flags set by base class
```

```
        pVer->dwFlags &= ~SF_NOTIFY_ORDER_MASK;

        //Set the flags we are interested in
        pVer->dwFlags |= SF_NOTIFY_OFDER_LOW | SF_NOTIFY_SECURE_PORT |
                                SF_NOTIFY_NONSECURE_PORT                                |
SF_NOTIFY_AUTHENTICATION;

        // Load description string
        TCHAR sz[SF_MAX_FILTER_DESC_LEN+1];
        ISAPIVERIFY(::LoadString(AfxGetResourceHandle(),IDS_FILTER,sz,
                                        SF_MAX_FILTER_DESC_LEN));
        _tcscpy(pVer->lpszFilterDesc,sz)
        return TRUE;

}

DWORD CAuthUserFilter::OnAuthentication(CHttpFilterContext* pCtxt
                        , PHTTP_FILTER_AUTHENT pAuthent)
{
        CString  oBuffer;                       // Buffer for client output
        DWORD    dwSize;                        // Size of the buffer.
```

```
LPVOID pvInOut;                    // Client input or output.

// See if we're gettint an anonymous request.
if(strlen(pAuthent->pszUser)==0)
{
        // Get the user's name.
        dwSize = 100;
        pvInOut = " ";
        pCtxt->GetServerVariable("REMOTE_USER", pvInOut, &dwSize);
        dwSize = strlen(LPTSTR(PvInOut));
        if (strncmp(LPTSTR(pvInOut), " ", 1) == 0)
        {
                // Set an error condition.
                pCtxt->ServerSupportFunction(SF_REQ_SEND_RESPONSE_HEADER,
                        "401 Access Denied",
                          NULL,
                          NULL);
        }
        else
        {
                // Indicate the user has supplied a password.
```

```
                pCtxt->ServerSupportFunction(SF_REQ_SEND_RESPONSE_HEADER,
                        NULL,
                        NULL,
                        NULL);

                // Store the user's name
                strncpy(pAuthent->pszUser, LPTSTR(pvInOut), dwSize);

                // Get and store the user's password.
                dwSize = 100;
                pvInOut = " ";
                pCtxt->GetServerVariable("AUTH_PASS", pvInOut, &dwSize);
                dwSize = strlen(LPTSTR(vInOut));
                strncpy(pAuthent->pszPassword, LPTSTR(pvInOut), dwSize);
            }
        }

        // return the appropriate status code
        return SF_STATUS_REQ_NEXT_NOTIFICATION;
}
```

```
// Do not edit the following lines, which are needed by ClassWizard.
#if 0
BEGIN_MESSAGE_MAP(CAuthUserFilter,CHttpFilter)
        //{{AFX_MSG_MAP(CAuthUserFilter)
        //}}AFX_MSG_MAP
END_MESSAGE_MAP()
#endif       // 0

//////////////////////////////////////////////////////////////////////
// If your extension will not use MFC, you'll need this code to make
// sure the extension objects can find the resource handle for the
// module,  If you convert your extension to not be dependent on MFC,
// remove the comments around the following AfxGetResourceHandle()
// and DllMain() functions, as well as the g_hInstance global.

/****

static HINSTANCE g_hInstance;

HINSTANCE AFXISAPI AfxGetResourceHandle()
{
```

```
    return g_hInstance;
}

BOOL WINAPI DllMain(HINSTANCE hInst, ULONG ulReason, LPVOID lpReserved)
{
    if(ulReason == DLL_PROCESS_ATTACH)
    {
        g_hInstance = hInst;
    }

    return TRUE;
}

****/
```

正如你所看到的，比起前一个 ISAPI **过滤器示例来，这儿的代码复杂多了。其中我们需要完整的一件工作是从过滤器直接向服务器传递信息，之后再把它传给客户机。这个过滤器完成了三步操作：**

1. **使用** GetServerVariable()**函数取得用户名，确定用户是否是匿名地访问服务器。（用户名返回值为空白，表明用户是匿名地访问服务器）。如果是，使用** SeverSupportFuction()**函数告诉服务器向客户机发出** 401 **错误信息。这使得显示出浏览器口令对话框，在这个对话框中用户输入口令字和用户名。**

2. 用户输入口令字和用户名之后，使用 GetServerVariable() 函数从输入流中提取它们，把用户名和口令字传递给 Windows NT 进行安全认证。

3. 使用 SF_STATUS_REQ_NEXT_NOTIFICATION 返回值告诉服务器，过滤器已成功地完成使命。

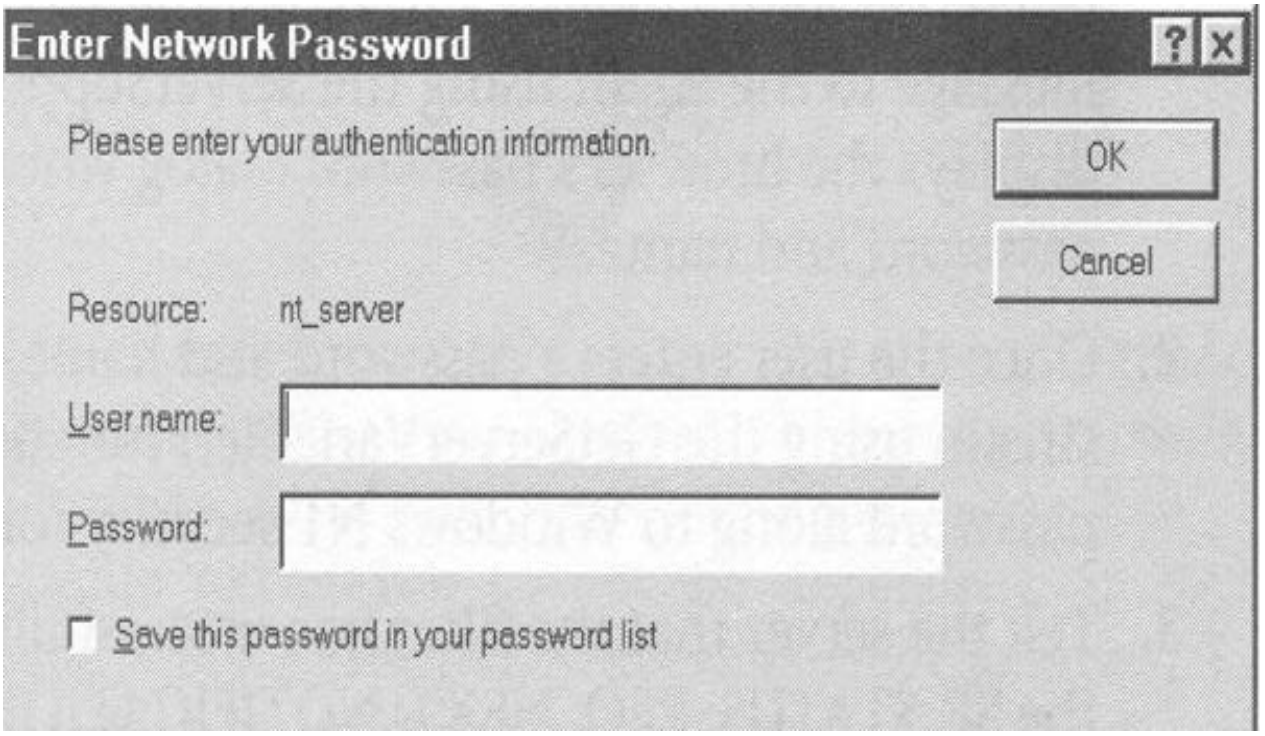
由于需要使用三个步骤，因此，使用安全过滤器时总有些令人吃惊的东西，例如，在使用 GetServerVariable()函数提取用户名与口令字和使用 Strncpy()函数把这一信息拷贝到 pAuthent 结构之间的时间内，对过滤器进行第二次操作期间将会进行解密。在第三个步骤中也要发生一些事情，例如，要对你需要直接与用户交互作出决定，这通常发生于对过滤器进行的第三次操作中。

请注意，我提供了 SF_STATUS_REQ_NEXT_NOTIFICATION 这个返回值。这是让服务器处理认证用户和显示 Web 页等细节的一个信号。如果想自己完成全部认证工作，就需要显示请求的信息。

技巧 IIS 通常在报错前提供三次机会让你正确地输入口令字。有两种办法可以防止这一行为的发生。第一种，在对过滤器进行的第二、第三次操作之间，自己完成认证。这样就可以给用户以另外的机会来提供正确的口令字，并将正确的用户名和口令字传递给服务器。第二种方法是使用 SF_STATUS_REQ_ERROR 返回值在第一次企图访问 Web 服务器失败后就终止进一步的访问企图。这一个返回值将显示一个服务器错误消息，并且可能会欺骗一些不那么精明的用户，让它们以为服务器发生了问题，从而不再企图访问服务器。

一旦编译了这个过滤器，使用本章第一个 ISAPI 过滤器相同的技术安装它。

运行这个示例就会看到如下图所示的口令对话框。



如果你键入了并不存在的用户名或口令字，则会看到如下图所示的服务器生成的报错消息。

输入一个有效的用户名和口令字就可以访问该 Web 站点了。

HTTP Error 401

401.1 Unauthorized: Logon Failed

This error indicates that the credentials passed to the server do not match the credentials required to log on to the server.

Please contact the Web server's administrator to verify that you have permission to access the requested resource.

第四部分

开发者眼中的

Visual C++



返回总目录

目　　录

第 14 章　安全性设计 4

14.1　精通 Windows 的安全 API 9

14.2　使用 Windows NT 的安全性 14

14.3　蛮荒不化的互联网 53

14.4　确保 Internet 下载代码的安全 60

14.5　安 全 标 准 71

第 15 章　建立帮助文件 86

15.1　确定要创建帮助文件的类型 89

15.2　组织帮助文件 92

15.3　使用 Microsoft Help Compiler 98

15.4　使用 Microsoft Help Workshop 121

15.5　为应用程序增加标准帮助 145

15.6　建立基于 HTML 的帮助文件集 152

15.7　向应用程序增加基于 HTML 的帮助 158

第 16 章　打包应用程序 171

16.1 **理解各种打包类型**.....173
16.2 **收 集 文 件**.....181
16.3 **建立安装程序**.....185

第 14 章　安全性设计

网络管理员在极力推行网络安全性时，将要面临很多问题，我们都知道：使用公用的或太短的保密字、用户不懂得安全操作以及对管理的完全忽视，都将直接威胁局域网的安全。幸运的是，由于有了类似 Windows NT 这样的操作系统，使得我们无论在本机还是在局域网内实施安全性时，都变得非常容易。学会如何使用这些安全特性并将它们置入你的应用程序中，对确保数据和应用程序的安全都是非常重要的。

在建立 Internet 站点或是 Intranet 站点时，安全性也是人们争论的主要焦点之一。公司可以通过建立一个 Internet 站点来获得公共的访问 ，也可以建立 Intranet 站点来尽量隐藏其身份，但现在很难知道哪一种会更好。不管你工作在什么样的环境中，市场上已经有很多公司开发的各类工具，以满足那些需要保护他们的数据不受损害的用户。然而，令人遗憾的是，当你建造自己的控件和应用程序时，所有这些华丽的工具实际上并不能从根本上保护你（当你用错误的方式使用它时，这些工具也不能很好地工作）。再次强调，学会 Windows NT 提供的安全特征，比如 ActiveX，将会帮助你提高远程访问技术的安全性。

在这一章中，你不应该去寻找那些定位在你特定工作环境中的具体答案。如果想在一章之中囊括当今所有的实用技术，或甚至囊括你曾遇上的每个安全问题，那都将是非常愚蠢可笑的。我们的重点是：从编程者的角度来看待安全问题，并讨论那些对编程人员实用的技术，但我们不针对具体的问题给出答案。

你最后所决定采用的方案应基于自己公司的需要以及你所使用的工具。

虽然 Internet 的安全性固然重要，但从整体上看，事实上，我们将更加集中精力关注 Windows NT 的安全性。我们也将关注那些使用远程访问所必须精通的技术，如 ActiveX 控件、分布式组件对象模型（DCOM）、组件对象模型加（COM+）等。这一章将要解答的一个问题是，编程人员可以采用什么手段来防止数据和应用程序代码受损？从编程者的角度来看，一个 ActiveX 控件不止是一个新的浏览器资源，它也可能是一个潜在的病毒源。在保护控件方面，除非是浏览器、编程者和 Internet 站点允许你更好地使用局域网资源，否则哪怕你做出了最仔细的努力，也将是枉费心机。这也意味着你将有可能因为一个笨拙的用户而暴露了自己的服务器，进而导致潜在的破坏。另外，因为 COM+还是一个海市蜃楼，所以你在使用这个新技术时应该特别小心。

微软为编写各类应用程序提供了各种 API 和开发工具，我们要做的第一件事就是花点时间学习一下这些 API，其中包括远程的和本地的，你至少应该看一点安全方面的 API，尤其是要看看类似 Windows NT 这样的产品所提供的本地安全性。既然你已经完全使用了操作系统，为何不充分利用它所提供的安全性呢？在一般的应用环境中，操作系统安全性带来的好处对任何人来说都是显而易见的（除非你正关注的是 Internet）。请牢记，你可以使用 ActiveX 控件来做一些事，如巧妙地处理注册登记或至少是用来询问用户和公司名称之类的信息，如果你可以使用 ActiveX 控件从注册中获取用户名和公司名，那么通过微软提供的操作系统服务，你也可以使用它来辅助实现其安全性。不难看出，你在 Internet 站点上所附加的那些安全性，将会分担很大一部分的安全压力。

我们在这一章要关注的另一个主题是：编程者在安全性方面所必须考虑的各类问题，也就是了解有谁进入了 Internet 站点、是如何进入的？了解你的组织结构中的脆弱部分，以及随意从 Internet 下载控件将会产生什么样的危害，这些都是非常重要的。要实施安全性，你首先应该知道你所保护的对象是谁、你所防止的东西又是什么？否则你的努力从一开始就是做无用功。

我们抛开了那些对 Internet 安全性的世俗顾虑后，将开始关注如何去实现它的安全性。这里有个问题，当 Sun 公司开始创建 Java 小应用程序（Java applets）时，它在安全性方面采用了一种称为“ Sandbox（沙箱）”的方法，即一个 Java 程序只能在它自己的 Sandbox 中使用，如果越过了 Sun 规定的这个严格区域，程序将不能访问操作系统，或者甚至不能访问硬件系统。为了使用 Java 小应用程序，你需要一些运行策略来确保访问是被严格控制的，这种方法的好处非常清楚：越少的访问意味着越少的安全顾虑。遗憾的是，它也严酷地束缚了 Java 处理工作的能力，这尤其表现在不同类型的数据要进行交换之时。

注　Java 在安全性方面采用了“ Sandbox ”的方法

Web 链接　即使 Java 也不是“防弹”（bulletproof）的，黑客已经发现了它执行代码中的漏洞，这使得 Java 也并非是一个完美的解决方案。例如，在 1996 年 3 月 22 日，在普林斯顿大学计算机科学系的 Drew Dean（ddean@ICS.Princeton.EDU）和 EDFelton（felten@ICS.Princeton.EDU），发现一个 Java 臭虫（bug），这个臭虫允许他们创建一个程序来删除用户本地盘的某个文件，他们通过 Netscape 的缓存机构下载一个文件到用户的机

器上，然后让 Java 来执行这个程序，并以此来嘲弄 Java，幸运的是，在 Netscape 的 2.01 版本中已经修复了这个漏洞。通过访问 http://www.cs.princeton.edu/sip/pub/secure96.html 站点，你可以找到有关 Java 安全问题的其它信息。

微软提出的另外一个方案是 ActiveX 控件，它被称作是“shrink-wrap”方法，使用 ActiveX，就像你可以使用软件的任何一个 shrink-wraped 片一样，你也可以完全访问操作系统。这意味着，你可以写出一些非常灵活的控件，他们可以充分利用系统提供的资源（这里也没有任何来自从访问数据方面的限制）。然而，从安全的角度来看，ActiveX 也能导致恶梦一样的恐怖，有什么东西能够保护你下载的控件，并防止它对你的系统进行破坏呢？微软还有另一个手段：厂商认证。当你在某个商店买下一个软件时，你便能知道它的制造公司，如果软件中有病毒甚至有险恶的臭虫，你也知道和谁联系，微软提出的 ActiveX 控件的认证方式也和你在商店买的软件一样，它的包装也确切地标识出了软件的生产厂家。

Web 链接 所有同时使用 VBScript 和 JavaScript 的开发厂家正持续不断地提高这两种脚本语言的安全性。不幸的是，要发现 VBScript 的未来发展方向还有点困难，通过访问 http://www.osf.org/~loverso/javascript/站点，你能够读阅读有关 JavaScript 的著名安全问题。

脚本语句是很容易被监视的，我们现在关注一下运行脚本语言的安全性。

不论是 VBScript 还是 JavaScript，他们的近期版本都不是非常安全的（虽然已做了许多工作来寻找 JavaScript 的漏洞）。下文介绍了三个最常见的问题。

欺骗用户上载某个文件。尽管 JavaScript 在上载一个文件时，必须征得用户的同意，但黑客却能够通过多种途径来绕过这种限定，黑客真正需要的只是一个含有趣味标题的按钮，对使用 Windows95/98 的用户来说，上载他们的用户保密字文件是非常容易的，这使得黑客闯进你的操作系统变得更加容易。

获得文件的目录。在上载你机器中的一个目录时，JavaScript 没有要求必须征得用户的同意，事实上，它也可以上载其它网络驱动器中的目录。黑客如果知道了你硬盘的组织结构，便能够更容易地闯入你的系统了。

跟踪访问的站点。黑客通过跟踪你访问的站点，便能够对你进行一些识别，而 JavaScript 使得黑客更容易这样做，它能够跟踪你访问的每个 URL，并将它们的地址发送到黑客的机器中。就像上载文件时出现的问题一样，用户必须给出许可，而黑客却几乎可以用任何一个东西来掩码这种许可。

本章也将关注一下安全标准，一些编程者把这些标准当成累赘，因为它们与编程没有直接的关系。我们并不在这些标准的全部内容上浪费时间，但是知道它们的存在以及如何找到它们，也是非常重要的。你们中许多人也都认为，没有什么理由要对一个已经安全实用的东西再作重新改造。如果有人已经发现了一种能防止访问中受到危害的方法，而且这种方法所有人都可以采用，那有何理由不采纳这个标准的方法呢？而这些也就是在安全标准这一节中所关注

的，它让你知道这些标准中什么东西有用，且让你学会如何去使用它们。

14.1 精通Windows的安全API

Windows NT 提供的安全级别对一个操作系统来说，几乎有点倾向于偏执，它允许你用很多的方式来设置安全性，包括既是日常用户又是文件级别的用户。你也可以创建用户组，并通过组来分配安全性，而不是对单独的用户来分配安全级别。另外，你也可以通过使用各种警报和登记文件来监视系统的安全情况。Windows NT 擅长于实时在地监视系统的活动情况，任何事件都逃不出其审查的范围，事实上，即便是从一个进程到另一个进程的信息传送，都要经过系统某一级别的审查。

对应用程序来说，Windows NT 提供的安全级别却是一把双刃剑。一方面，应用程序不能做太大的破坏，大多数情况下，在造成安全破坏之前，Windows NT 便能够简单地中断错误的应用程序；另一方面，对一些在 Windows95/98 中运行很好的应用程序，施加这么严格的安全限制，将会对程序的运行造成破坏。Windows NT 提供的安全性，从本质上说，确实影响了你的机器本身所能提供的兼容性。

技巧 如果只是想设计一个一般的应用程序，而没有什么与安全性相关的服务，你也许潜意识想用 Windows95/98 来做。可另一方面，Windows NT 提供了那么多的安全服务，以至于你可以创建一个界面几乎不能让人接受的应用程序，然而，要付出的代价是，你的应用将不能在许多机

器上运行，因为 Windows NT 和 Windows 95 在安全性上几乎处于两种完全不同的极端——Windows NT 具备了几乎太多的安全性，而 Window 95 却几乎没有。

Windows NT 提供的大量安全特性使 Internet 站点和用户得到了极大的便利，它的本地安全性使 Windows NT 成为了 Web 站点中的优秀平台。另外，如果你在使用 Windows NT 提供的标准特性来编写代码时遇上了什么麻烦，你也可以改进这些特性。例如，在第 13 章中，你能编写一个 ISAPI 过滤器来监视后台的事件，如果将 ISAPI 过滤器和 Windows NT 的安全特性结合起来，便能够提供至少在跟踪重要安全事件中所需要的任何东西。也就是说，即便你不能阻止黑客对你的安全性进行破坏，你至少还能够跟踪黑客做了些什么，以将其破坏减至最小。另外，识别黑客闯入的地方，也将是提高安全性的一种方法。

安全特性并不局限于 ISAPI 过滤器或桌面应用，你也能够将 Windows NT 的安全性纳入 ActiveX 控件或扩展 ISAPI 中。例如，你可以编写一个 ActiveX 控件，并用它来检测登录你的系统的用户是否真正获得了访问许可。你甚至还可以编写一个扩展 ISAPI，它可以根据系统允许用户的不同访问级别，发送给用户不同类型的 HTML 网页。

从理论上说，你也可以将 Windows NT 的安全性加入 DLLs 中，通过 DCOM 的使用，它们能被客户机器所访问（甚至可以是 Windows 95/98 的客户）。因为 DLL 将在 Windows NT 服务器上运行，而不是像常规一样下载到客户端，所以客户实际上受到了服务器安全特性的保护。实践证明，在应用中加入此类安全特性，并不增加多少网络负载和服务器处理时间。相比之下，你当然宁愿通

过 DCOM 实施安全性。

既然 Windows NT 已经提供了这么多的安全性，那么为什么还要在你的应用中努力去创建附加的安全性呢？第一个原因是，目前为 Internet 安全性设计的 API 标准还存在着漏洞，例如，我们已经说过的 Internet 部件下载服务的漏洞（在第 8 章中，我们可以看到 Internet 下载过程的部分论述），因为当前市场上的各种浏览器提供的安全防线还存在着漏洞，所以一些神通广大的人可以通过各种手段绕过我们的安全防线（我们将集中讲述 Internet Explorer，当然 Netscape 和其它浏览器也肯定存在这样或那样的毛病）。这些安全漏洞并非存在于 API 本身，而是存在于过去为 Internet 设计的一些解决方案中。实质上，这些过去设计的解决方案到现在已经过时了。以下将告诉你 Internet 部件下载服务中的三个漏洞。（当然，肯定还有其它漏洞）：

HTML<A HREF>标志。人们有办法通过使用<A HREF>标志来下载和运行一个可执行文件，在 Internet Explorer 的 3.x 或 4.x 版本中，对代码的检测过程中存在这个问题，它当前的做法是，使用 URL 名字直接下载代码，然后调用 WinVerifyTrust 来检测代码的合法性（这与第 8 章中的描述一样），这种方法 100%安全吗？回答是否定的，因为你使用了标准进程以外的一些东西来检验文件的内容，在这里你将依赖于这个 HTML 分析器。

脚本。现在的脚本对所有安全检测来说，是完全自由的。没有什么方法能检测出是谁创建了这些脚本，或它将会对你的机器做些什么？更严重的是，没有什么方法能检测出这些脚本将会从你的机器中取回些什么信

息！（在这一章开始，我们看到了 JavaScript 能从你的机器中取回的两种信息：用户硬盘中的目录和用户访问的站点）。微软正在创造一些脚本的认证，一旦脚本认证变成了现实，浏览器便可在运行代码前先调用 WinVerifyTrust 来检测这些脚本。

完整应用程序的下载，或其它更复杂的下载情况。IE 目前在为检测特殊类型的下载，做了许多很好的工作，例如，在下载一个 OCX 时将插入一个 WinVerifyTrust 进程。如果下载的参数超出了 IE 的检测限制范围，将会发生什么情况呢？例如，某一用户可能想下载并安装一个 Doom 或其它的游戏，而在安装中可能发生一些不可预料的事情，如申请注册或重启系统等，而此时的 IE 却不能控制它。微软正打算在以后组件下载版本中加强其健壮性，使它能控制这类事件。

Web 链接 作为保护策略中的一部分，你也应该测试多种浏览器并检测它们的反应（尤其在你创建一个公共访问站点时）。编程人员在与多种产品一起工作时，应该在头脑中形成一种基本观念，那就是不断地跟踪你访问过的所有 Web 站点，一种名叫 NavEx 的产品允许你将 IE 的收藏站点复制成 Netscape 的书签，反之，它也能将 Netscape 的书签复制成 IE 的收藏站点，通过访问 http://mach5.ocs.drexel.edu/navex/，你可以下载到这个产品。

注释 在这一章，我们也将探讨 Internet 技术中的其它漏洞。例如，表 14.1

列出了一些为填补这些漏洞而设计的新标准。如果没有出现安全问题，SHTTP、S/WAN 和其它类似的技术将是不必要的。MIME 有一种新技术叫 S/MIME，它能确保其它人无法偷窥你的电子邮件（mail）。

为了精通 Windows NT 是如何增强 Internet 站点安全性的，我们首先要来看看操作系统本身提供的安全特性。Windows NT 提供的所有安全特性，如映射网络驱动器，都能被应用程序所访问，许多编程人员好象忘记了，使用 ActiveX 控件也能够访问这些特性。例如，你可以像在第 10 章中创建一个按钮一样来调用它，其目的是显示一个网络映射对话框，以供用户选择一个驱动器来进行网络映射。

当然，Windows NT 也有一些安全特性并没有涉及 Internet 的安全领域。如映射网络驱动器，在应用程序中你可以通过创建一个按钮来调用它，但它并没有 Internet 的安全保护。然而，如果你在一个 ActiveX 控件中增加上 Windows NT 的一些其它安全特性，则可以实现它的 Internet 安全性。例如，你可以在一个 ActiveX 控件中增加上 Windows NT 的口令保护，这样，用户每次想要访问你的站点时，将显示一个登录对话框，要求用户输入用户名和口令。

在一个 Internet 的应用程序中，通过 ActiveX 控件来显示登录对话框，实际上完成了两件事：第一，Windows NT 提供口令保护的安全性比 Windows 95/98 要强，黑客如果没有猜出正确的保密字，将要周旋很长时间才能通过这个对话框；第二，你可以让服务器登记下所有的访问记录——请谨记，Windows NT 提供给了你监视任何事件的能力。万一实在有人闯入了你的系统，你至少也可以知道这个黑客使用了哪个帐户，并根据这个帐户的安全等级，评估黑客对你

的系统可能造成的破坏程度。

Windows NT 提供的安全性 API 函数将帮助你创建一个更加安全的 Internet 环境。要使用这些安全特性，在客户站点端，你可以使用 ActiveX 控件，而在服务器端，你可以使用扩展的 ISAPI，并通过在后台运行的 ISAPI 过滤器来监视事件的安全性。

14.2 使用 Windows NT 的安全性

在上一节，我们已经阐述了推荐使用 Windows NT 做 Web 服务器的一些理由，其最重要的一点是，能够使用 Windows NT 的安全性 API。然而，懂得为什么要存在 Windows NT 的安全性 API 只是一个起步，现在，我们将要关注的是，在现实中你是如何实施这些安全特性的。在这一章中，我们实际上讲述了五种不同类型或等级的安全特性，下面将对每种类型进行定义：

注 使用 Windows NT 作为 Internet 服务器的一个主要原因是，你能使用它的安全性 API。

内置安全性 作为一个操作系统，Windows NT 有一定级别的安全性。在 Windows NT 中，每个对象都对应着一定的安全性，我们将关注这些安全性的实施方法。使用对象级别的安全性意味着：如果没有正确的授权，几乎没人能有机会访问操作系统的任何部分或任何数据，当然，也不是一点机会也没有——我们总是假设你的安全性不是最完美的。

私有通讯技术（PCT） PCT 是微软和 IETF 一起创建的一种特殊的 Internet

安全性技术。在近期版本中，PCT 将允许客户和服务器建立私有的通讯，以保证通讯不会被人窃听到，这种安全性的实现依赖于数字签名和加密方法。

Windows NT 通过 HTTP 进行身份认证 许多人都错误地认为，使用这些安全性来工作将是非常繁琐和复杂的，但有时候，看似烦杂的东西却可以通过一个简单的方案做到尽善尽美。我们从 Internet 的角度来看，在 Windows NT 中进行身份认证有两种形式，一种形式是简单地询问用户的用户名和口令，并和服务器内的访问列表进行对照，这种形式又可通过两种方法实现，一种是在对 Windows NT 进行标准登录时进行的询问和回答；另一种则基于当前会话的设置，它也要求客户提供必要的用户名和口令，或者说，让用户使用客户提供的信息登录到机器上。另一种形式相对简单些，它依赖于一种叫安全套接层（SSL）的技术，SSL 又是依靠加密和数字认证来工作的。如果你过于担心安全性问题，你甚至可以将这两种形式结合起来使用——它们彼此间并没有什么相互的制约关系。

数字签名 它是指发送人对他发送的文档或可执行文件进行签名，通过检测签名你可以判断其真伪。数字签名用一系列的私有或公开密钥来实现这个安全等级。

加密 API 有些类型的安全性依靠多个保护层来工作，也就是说，如果一个黑客闯过了你的第一个安全保护层，则另一个保护层将阻止他进一步的访问。这就是加密 API（也叫 CryptoAPI）所有要做的，它是一个附

加的保护层，附加在所有已经存在的保护层上，它能保护你免受黑客更深层次的攻击。

在本章的下几节中，我们将阅读到几种新的技术，你将发现它既嵌入在 Windows NT 中，也嵌入在新型的浏览器中。作为 Internet 服务器中的一种，Windows NT 支持如微软的 Internet 信息服务或对等网信息服务（在 Windows NT4.0 和 Windows95/98 的 OSR2 版本中免费提供）。我们将有趣地发现：和新版本的 Netscape 转接器一样，微软的 IE3.x/4.x 中也提供了内置 SSL 的支持；你也将发现，PCT 至少也支持 IE。我们也将关注数字认证码（即数字签名）技术，现在的许多软件厂家已经采用了它，它允许编程人员对其作品进行数字化签名，当你下载这些经过数字化签名的程序时，程序代码在签名后的任何损坏都将显示出来。它带来的好处是，你不会轻易下载到一个病毒感染过的程序。

Web 链接 通过访问 http://www.microsoft.com/iis/guide/pws.asp?A=2&B=5，你可以为你的 Windows 95 下载某个版本的个人 Web 服务器（PWS，Person Web Server）（Windows 98 的安装包中自带了 PWS），这个版本的 PWS 也与 Windows 95 的 OSR2 版本以及微软的 Windows NT 可选包随同销售（Windows 95/98 也有其可选包）。虽然 PWS 不会去满足制造 Web 站点的需求，但它能够让开发者很好地检测 Internet 应用程序。要知道，因为 Windows 95/98 没有提供与 Windows NT 一样的安全性，所以，如果使用 Windows95/98 的 PWS 版本，你将检测不到应用程序的任何安全特性。（幸运的是，在与 Windows NT 4.0 Workstation 一

同销售的 PWS 中，将允许你完全检测这些安全特性）。

加密也是保护你的数据的另一个可行方法，让我们参看一下下述微软已经开发的加密函数 CryptoAPI，这部分的技术至少是在 Internet Explorer 内的一种主要形式。然而，不管你现在能够使用它做什么工作，CryptoAPI 确实在程序内工作着，另外，对 CryptoAPI 的开发环境进行扩展，只在美国和加拿大是有效的。

内置的安全特性

在安全性方面，Windows NT 几乎是当前市场上最超前的操作系统。如果你对微软的这种地位还有什么怀疑的话，那么请去看看 Windows NT 获得的各种证书吧。你可以通过使用它的安全性能来获取很多便利，例如，你可以在一个 OCX 中使用这些高级功能，或通过 Internet 获得一些特殊功能的访问。在做这些事时，你必须受到一些限制，以免破坏你的网络安全。使用一些来自 Internet 的特性是不切实际的，因为它们并不重要。

幸运的是，ActiveX 控件却不必对 Internet 进行限制，没有什么条令规定，你不能在你的应用程序中创建一个有特殊用途的控件。在数据库或其它应用程序中的 ActiveX 控件，能被网络管理员（或其他有相同权利的人）正常地使用，这使应用程序的工作变得更加简单和可靠。事实上，如果你需要的话，ActiveX 控件也可以设计为多种个性化用途，你甚至可以给它增加检测控件当前位置的能力，或增加一个指定的位置作为特性页面配置的一部分，一些人可以选择其

用于 Internet 的特性子集，而另外一些人则可以选择其用于局域网（LAN）的特性子集，还有一些特性集则是用于本地的。

也许你们中许多人会问：需要什么类型的管理员来管理安全性，且不需使用网络操作系统（NOS）附加的工具呢？Windows NT 确实提供了大量的管理工具，而且这些工具都非常容易使用，所以一些编程者认为，再附加一些工具是不值得的。这个问题不好回答，但是，当大部分人在考虑 Window NT 下的安全性时，几乎总是这么想：如果管理应用程序的人不是网络管理员，而是练习使用网络操作系统的某个人，这又该如何处理呢？我们说这个人可能是一个工作组管理员或其他独立的个体，他没有必要看到整个网络图，而只需要足够的信息来维护他所负责的应用程序，你将发现这种情况是经常发生的，尤其是一个拥有许多小工作组的大公司，更是属于此类情况。尽管网络管理员不懂得正确管理应用程序，但也不愿让工作组管理员自由地访问整个网络。

注　ActiveX 在局域网环境的一个最好的应用是，它提供一定的子网安全管理能力。

不管你是在为 Internet、还是为局域网（LAN）或是为广域网（WAN）创建控件，你都将发现，懂得网络底层的安全构造是非常重要的。Windows 95/98 没有提供像 Windows NT 一样多的安全特性，所以，在 Windows 95/98 中，你有时会发现自己没有增加上应用程序的安全性。然而，尽管 Windows 95/98 没有为应用程序提供某个安全特性，但有时它们和 Windows NT 使用同样的安装程序，所以应用程序安全模块将同时为这两种操作系统工作（在其它情形下，你可以干脆为 Windows NT 使用单独的安全模块，以增强其安全性能，请看下

面的具体注释）。

注释 与 Windows 95/98 相比，Windows NT 确实提供了更多的安全 API 函数，因为它的安全性更加强壮。现实中，因为 Windows 95/98 缺少对 Windows NT 安全函数的支持，所以在使用它时，你会经常遇到一些障碍。例如，在 Windows 95/98 下，你不能调用 GetUserObjectSecurity 函数；而在下一节中，我们将会看到的操作访问令牌的函数，也大多不能在 Windows 95/98 中使用。弄清一个函数是否被支持的最好方法就是直接检测它，如果你在调用它时收到一个“ERROR_CALL_NOT_IMPLEMENTED(value 120)”的信息，则你便知道它只能在 Windows NT 中使用了。

在 Windows NT 和 Windows 95/98 中，“对象”（Object）这一名词的使用更加宽松。我们说，在事物的背后确实潜伏着许多对象，但你可能发现，“对象”这个名词的定义已经不能完全符合 C++的习俗了。在下几节中，我们把对象大体看成是：在运行一个指定的安全任务时所需的代码和数据的总和。或者说，每个安全对象自身包含了一个为充当某个角色而设计的单元。（在 Windows NT 和 Windows 95/98 的许多地方，微软主要选用 C++中“对象”的定义，因为作为 MFC 的一部分，它提供了所需的机能。然而，在阅读本章或微软的其它文档时，你不应该把对象看成是一个严格的 C++对象，而应该多加点 COM 的意识）。

知道了所有东西都是对象后，我们将更容易理解安全性——这至少是一个起点，当然，对象本身也就是一个起点。对象是被用户所访问的，所以在 Windows

中，安全性是指将对象所受的保护与用户的访问权进行比较。如果用户拥有足够的访问权（访问权等于或超过对象所受的保护），则用户能够使用这个对象。在 Windows 的文档中将对象所受保护的级别称作安全描述符（security descriptor），这是一种结构，它能告诉安全系统：用户需要什么样的权力才能访问这个对象；而用户则有一个访问令牌（Access Token），它是另一种结构，它能告诉安全系统：在一个给定的位置，用户有什么样的访问权。用户将访问令牌交给 Windows NT 系统，并以此获得对对象的访问。（如果把访问对象看作是一辆公共汽车，把 Windows NT 看作是驾驶员，则“车票”是访问令牌的一个很好的比喻，乘客只有出示车票才能搭车）。图 14.1 描述了这两种结构。

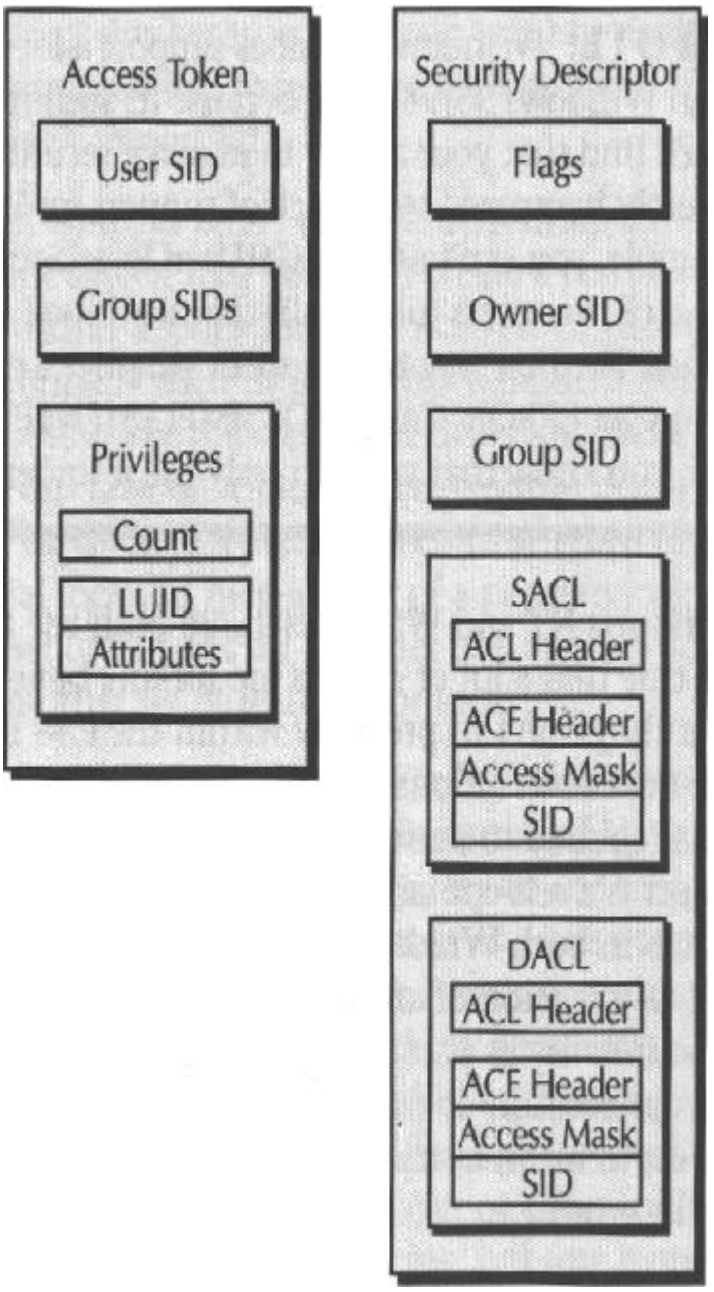


图 14.1　**访问令牌定义了用户的权利，而安全描述符则定义一个进程的保护级别**

以上，你只是稍微了解了一点 Windows 95/98 **或** Windows NT **的安全性，简单地知道了它有安全对象和用户访问令牌，这与帮助你理解和使用** Windows

安全 API 函数还相差很远。在下面几节中，我们将细致地学习访问令牌是如何工作的；我们也将关注一下安全描述符。如果你只是对 Internet 感兴趣的话，在你使用 ActiveX 实施安全性时，可以完全不必懂得这些知识。然而，学会它，将会帮助你设计出更加符合心意的 ActiveX 控件。

理解访问令牌

在 Windows 中，你将发现有两种方法可以查看用户的权限，而且两者都是通过这样或那样的形式与对象关联的。用户的访问令牌有一个安全标识符（SID），它在整个网络中识别用户——它就像是一个帐户号。安全识别符表示了用户归属的组和拥有的优先权。每个组也有一个安全识别符，所以，用户的安全识别符也包含了他所属各个组的 SID，并将引用这些 SID。在 Windows NT 下，要想改变用户访问令牌的内容，你应该正规地使用用户管理器。

注　Windows NT 支持两种类型的安全标识符（SID）：用户 SID 和组 SID

那么访问令牌的优先权是指什么呢？首先是用户拥有特权数——不是用户所归属的组的数目，而是指访问令牌内指定的特权入口数，这部分还包括特权入口的阵列。每个特权入口还包括一个局部唯一的标识符（LUID）和一个属性掩码（Atrribute mask）。LUID 实质上是一个指向对象的指针，而属性掩码能表述用户对对象的权力。组的安全识别符也与此雷同，它们包括一个特权计数和特权入口的阵列。

技巧　现在，你可以去看看 Visual C++软件提供的 Windows API 帮助，找一找与安全标识符或访问令牌相关的 API 函数。例如，与 SID 相关的

函数包括 CopySID 和 AllocateAndInitializeSID；你会发现 OpenProcessToken 和 GetTokenInformation 函数也是非常重要的，它能在任何语言下保证应用程序安全正常工作。

高级技巧

观察访问权限传递

你还该知道的一件事是：有些类型的对象，如果它的权限没有被其它 SID 终止，则可能会继承到它的最低节点。例如，如果你让一个用户对硬盘的“\Temp”目录有读写的权限，则用户对该目录的子目录“\Temp\Stuff”也有相同的权限，除非你分配给这个子目录别的权限。对于容器对象也是如此，如果分配一个用户对容器对象的权限，如一个 Word 文档，则它允许用户查看这个容器下的所有内容，很多情况下甚至还有别的文件。由此看来，因为你可能会不经意地分配给用户过多的权限，所以使用安全审察来跟踪用户对服务器中各类对象的确切权限，是非常重要的。

使用访问令牌

因为访问令牌函数是你实施安全性前必须了解的第一个难点，下边，就让我们简要地谈论一下 Windows 提供的访问令牌函数。对用户的帐户做任何工作，哪怕是找出谁在访问指定的工作站，都必须了解访问令牌。正如前面所讲

的，访问令牌是安全等式用户方的中心部分。在对一个用户的帐户进行访问时，你几乎总要用到的一个函数是 OpenProcessToken，请注意这个函数的名字，它能够处理的对象可以是任何类型的进程、线程或用户等等，该函数的作用是获取附加在这个进程上的令牌句柄。例如，如果你查询一个用户的帐户，你必须有 TOKEN_QUERY 优先权，（你的访问令牌中必须包括系统委托的一些必要的权限，这就是为什么管理员能访问某个令牌，而其它用户却不能访问的原因）。而对用户的帐户要做任何改动时，你必须拥有 TOKEN_ADJUST_PRIVILEGES 优先权。还有许多其它的访问权限，我们在这里不再一一表述了。

一旦你拥有了一个访问令牌的句柄，你需要决定将对它做些什么。如果你想改变用户做某事的优先权，则你将需要这个优先权的局部唯一标识符（LUID），所有这些优先权在 WINNT.H 文件中都是以“SE_”开头的。例如，SE_SYSTEM_PROFILE_NAME 优先权允许你收集这个系统的配置信息。有些 SE 的值与用户无关（例如，SE_LOCK_MEMORY_NAME 特权允许进程锁定内存中的物理页面）。通过调用 LookupPrivilegeValue 函数，你可以获取对应局部系统上的优先权的 LUID。一般而言，你将使用 AdjustTokenPrivileges 函数来实现你所需要的修改。

注　在 WINNT.H 文件中，查看一下 Windows NT 定义的优先权列表，这些优先权是你能修改的。

查询用户帐户（或访问令牌的其它信息）是直接了当的。你可以调用 GetTokenInformation 函数来获取你需要的信息，这个函数需要一个令牌类参数，它把你所需信息的类型告诉 Windows 系统。例如，如果你想知道一个指定用户

的信息，你应该使用 TokenUser 类参数，你也将需要提供相应的结构，以让 Windows 保存你请求的信息，这一点与基于令牌类的请求不一样。

了解安全描述符

现在让我们来关注一下安全描述符。在图 14.1 中，每个安全描述符包括五个主要的部分。第一部分是一个标志列表，这些标志指明了描述符的版本号、格式和访问控制列表（ACL）的状态。

接下来的两部分包含了多个安全标识符（SID），所有者标识符表述了对象的拥有者，这不必是一个单独的用户；Windows 也允许你在这里使用组的 SID。有一点是受限制的，即组的 SID 必须出现在想要改变入口的人的访问令牌中。组标识符允许一组人拥有某个对象。这两种标识符中，只有所有者标识符在 Windows 下是重要的。组的标识符作为一个部分用于 Macintosh 和 POSIX 的安全环境。

最后两部分包含了各种访问控制列表（ACL）。安全访问控制列表（SACL）控制了 Windows 的审计功能，用户或组每次访问一个对象时，这个审计功能便启动了，Windows 在审计记录中产生一个登记项；全权访问控制列表（DACL）控制了谁能真正使用某个对象。你可以将一个组或用户分配给一个指定的对象。

注释 实际上，有两种型号的安全描述符：绝对安全描述符和自身安全关联描述符。绝对安全描述符在它的结构中包含了每个 ACL 的真正拷贝，这种类型的安全描述符用于需要指定句柄的对象；而自身关联安全描述符只包括了指向 SACL 和 DACL 的指针，这种描述符在改变组的

权限时，节省了内存并减少了所需时间，当某个特定组内，所有对象需要同一级别的安全性时，你应该使用它来处理，例如，通过这种方式，你能将单个进程内的所有线程赋予安全性。在你要保存安全描述符或将它传送给别的进程前，Windows 要求你把一个自身安全关联描述符转换成绝对安全描述符。通过 API 函数获取的描述符都是自身关联类型的，你必须在保存它之前将其转换。通过对 MakeAbsoluteSD 和 MakeSelfRelativeSD 两个 API 函数的调用，你可以在这两种类型的描述符之间转换。

一个 ACL 由两种类型的表项组成：第一个表项是个结构头，它列出了 ACL 包含的访问控制表项（ACES）个数，Windows 使用这个数字检测是否到达了 ACE 列表的末端（这里没有任何其它记录或手段能够检测每个 ACE 的准确大小）；第二个表项是个 ACE 数组。

警告 不要手工直接修改 ACL 或 SID 的内容，因为在以后的 Windows 版本中，微软可能改变它们的结构。Windows API 为改变这些结构的内容提供了丰富的功能。如果要对某种结构进行任何的改变，请通过调用 API 函数来实现，以免不必要的冲突。

那么什么是访问控制项呢（ACE）？ACE 定义了单个用户或用户组在一个对象上的权限。每个 ACE 由三部分组成：首先是一个结构头，它定义了这个 ACE 的类型、大小和一些标志；接着是一个访问标志，它定义了一个用户或组对这

个对象的权限；最后是用户或组的安全访问标识符（SID）的项。

一共有四类 ACE 头（其中有三类已用于当前的 Windows 版本中）。其中，access-allowed 类出现在 DACL 中，它认可用户的权限，你可以使用它增加用户对一个对象的权限。例如，虽然你想通过禁止用户修改系统时间，以使网络上所有机器的时间同步，但白天还是可以修改时间，而用户需要有这个权限，你可以通过使用一个 access-allowed 类的 ACE 来给予用户改变时间的权限；另一类是 access-denied 类，它废除用户对某一对象的权限，在一个指定的系统事件中，你能够使用它来拒绝对某一对象的访问，例如，当你对某一远程终端进行某些更新时，你可以取消其他人对它的访问权限；还有一类是 system-audit 类 ACE，它与 SACL 一起作用，它定义了指定的用户对哪些事件进行审查。当前还未使用的是系统警报 ACE，它允许 SACL 或 DACL 设置一个指定事件发生时的警报。

注 在当前的 Windows NT 版本中，使用的三类访问控制项（ACE）是 access-allowed、access-denied 和 system audit。

技巧 现在，你可以翻阅一下 Windows API 的帮助文件，看看 Windows 提供了什么类型的访问权限。为了获得这些信息，你应该更关注各种类结构，尤其是 ACL 和 ACE 的结构。看看 ACE 标志，它决定了在一个容器下的对象是如何作用的。例如，查一下 CONTAINER_INHERIT_ACE 常数，它允许子目录继承父目录的安全特性。

使用安全描述符

到现在，你只是了解了什么是安全描述符，以及它包含的各类结构是如何相互作用的，你还应该知道怎样开始一个实际的访问进程，并使用安全描述符来写一个程序。你必须首先了解的是：安全描述符不同于访问令牌，它没有普及开，所以你不能使用一个标准的函数集来访问它们。实际上，一共有五类安全描述符，每类安全描述符都使用不同的函数集来访问对象。（你必须拥有 SE_SECURITY_NAME 特权，才能使用这些函数）。

注 Windows NT 支持五类安全描述符，每类都需要不同的函数集来访问描述符

文件、目录、管道和邮件槽。通过使用 GetFileSecurity 和 SetFileSecurity 函数，可以访问这种类型的对象：

注释 只有 Windows NT 下的 NTFS 文件系统提供了安全特性。Windows 95/98 下的 VFAT 文件系统只提供了很少的安全特性。你不能从 HPFS 或 FAT 文件系统中分配或获取安全描述符。FAT 文件系统没有提供任何扩展属性的空间，而这些空间是增加安全性所必须的；HPFS 文件系统虽然提供了扩展属性，但它们没有任何安全特性。在上述的这些文件系统中，NTFS 是最安全的。然而，不要错误地认为会有完全安全的文件系统。在 Internet 上，有一些应用程序能够读取 NTFS 分区的内容，即便用户没有正确地登录到 Window NT 中。

要访问进程、线程、访问令牌和同步对象，你必须使用

GetKernelObjectSecurity 和 SetKernelObjectSecurity 函数。所有的这些对象都是内核对象，为了更好地保护这些对象，它们也都有自己的安全描述符。

要访问 Windows 站点、桌面、窗口和菜单，你可以使用 GetUserObjectSecurity 和 SetUserObjectSecurity 函数。一个 Windows 站点包括键盘、鼠标和屏幕，即你用来访问系统的所有硬件。桌面包括窗口和菜单。这四个对象按此次序依次从前一个对象继承权限，也就是说，桌面将继承站点的权限。

要访问系统注册键，你必须使用 RegGetKeySecurity 和 RegSetKeySecurity 函数。注意，Windows 提供的所有注册类的函数都是以 Reg 开头的。

可执行服务对象，QueryServiceObjectSecurity 和 SetServiceObjectSecurity 函数用于可执行的服务对象。奇怪的是，在 Windows 的 API 帮助文件中，这两个函数都没有出现在其它安全函数的引用中。你在寻找这些函数之前必须已知道有这么两个函数。可执行服务是 Windows 提供的后台任务，如 UPS 的监视功能。通过双击控制面板中的服务程序，你将找到你的系统所支持的服务。

一旦你获得对某个对象的访问权限，你将发现，通过使用一般的 API 函数集，你便能够处理很多任务。例如，GetSecurityDescriptorDACL 能够从任何类的描述符中获取一份 DACL 的拷贝。换句话说，这些对象的描述符跟随一个同样的格式——即便大部分成分的描述符长度不同。导制这些长度不同的原因之一是，它们各自包含了不同数量的访问控制项（ACE），另外，安全标识符（SID）

的长度也不同。

要查询或修改一个安全描述符的下一步工作是分离这些成分。例如，通过使用 GETACE API 函数，你能够查看在一个 DACL 或 SACL 中单独的 ACE；通过使用与 SID 相关的函数，你也可以使用所有者标识符或组标识符（我们已经在本章的访问令牌部分阐述了这些函数）。可以满意地说，只要按指定的步骤做，你便能够使用一般的函数集来操作安全描述符。大致上说，对安全描述符的任何访问，一般都要经过下述三个步骤：

1. 获取描述符
2. 删除指定的成分
3. 修改指定成分的内容

要修改安全描述符，你应该按相反方向执行这些步骤。换而言之，你先使用类似 AddACE 函数来增加一个新的 ACE 至一个 ACL；然后，使用 SetSecurityDescriptorSACL 来改变一个描述符中的 SACL；最后，使用类似 SetFileSecurity 的函数（假设你想要修改一个文件对象）来保存安全描述符。

Windows 中的 ACEing 安全性

当你开始考虑 Windows 在 DACL 中计算 ACE 的方法时，你也许会发现一些潜在的存在问题的地方：虽然 Windows 的系统应用程序会自动关注这些问题，但你必须在自己应用程序中编制好了，才能运算出同样的结果。（SACL 也有同样的问题，但它只是影响审查，从系统安全的角度看，它不是那么要求苛刻的）。

Windows 是按照 ACES 在 ACL 中出现的次序来计算访问控制结果的。首

先，它可能不是一个很好算法，而且，在某些情况下它还会产生一些问题。例如，如果你想废除所有用户对某一区域的权限，但是在他或她的访问控制列表中包含了一个能够访问这一区域的组（即他或她是这一组的成员），这种情况的结果会如何呢？如果你将 access-allowed 访问项放在列表的第一位，则这个用户将能够访问这一区域——一旦 Windows 发现第一个满足用户所需权限的 ACE 时（或者一个禁止所需的权限的 ACE 时），它将停止继续搜寻。许可的权限是累加的，如果一个 ACE 允许对一个文件进行读的权限，而另一个允许写的权限，则当用户询问读和写的权限时，Windows 将综合这两个 ACE 并允许用户进行读写。下面的注中表示你应该将所有 access-denied 访问控制项放置列表前端，以防止安全上的潜在漏洞。

注 ACE 在 ACL 中排列的先后次序是非常重要的，因为 Windows 按次序对它们求值，并一旦发现第一个 ACE 允许或拒绝对某一对象的访问时，便停止继续搜索。

你也必须注意组的安全标识符（SID）的排列次序，因为用户从他（或她）所属的组获取的权限是相累加的。这意味着，如果一个用户属于两个组，其中一个组对某个文件有访问的权限而另一个没有，假如允许访问的组在列表中排列在前，那么用户可以访问这个文件，否则不能。

显而易见，为了将组排列成最好的次序，你将花费很多的时间。当用户拥有的组的权限和独立的权限数增加时，则可能出现无意识的安全漏洞。所以，仔细地创建一个组并严格限制用户单独的权限是非常重要的。

其它安全考虑

当你关注在 Windows 95/98 或 Windows NT 下的安全性时，还有两方面需要考虑：数据保护和服务器保护。前者对付客户的这种能力：当客户通过服务器（这里不是指文件服务器，而是某些类型的 DDE 或其它应用服务。）访问数据时，访问了那些没有获准访问的数据。我们用这种方式思考一下：如果客户对某一指定类型的数据没有访问权限，但是通过调用他有权访问的 DDE 服务器访问了这些数据，这将会有什么后果呢？服务器又应该如何保护它们的数据，而不是被迫成为造成安全破坏的同谋呢？

Windows 提供了一些 API 函数允许服务器模仿一个客户。大体上是：为了检测出客户是否对一个数据或进程有过多的权限，这些函数允许一个服务器假扮成客户所受的安全限制。例如，Windows 的一个 Word 用户可能需要访问 Excel 的一个数据文件，而这个用户可以通过 DDE 来获得这个文件的访问，在这种情况下，服务器需要检测一下这个用户是否有权访问这个文件，然后才传送所需的数据。当用户使用这种技术时，服务器甚至能找出客户的优先权。服务器唯一关心的是对它所管理的数据、资源和环境的保护。

这个 API 函数集支持三种不同类型的通信：DDE、命名管道和 RPCS。对每种通信类型，你必须使用不同的 API 函数。例如，要模仿一个 DDE 客户，你应该使用 DDEImpersonateClient 函数。Windows 目前在模仿这方面所提供的支持还有一些限制。例如，它现在还不支持 TCP/IP 连接，所以在这种情况下，你必须依靠别的手段来检测一个用户是否有合适的访问权限。

另一个安全考虑是如何保护服务器自身。在一个 Windows Word 用户调用

Excel 服务时，用什么方法来防止这个用户对 Excel 服务器本身进行破坏呢？请确信，对文件和其它有名字的结构，服务器自动会附加上一个安全描述符到这些对象上，并进行严格的保护（一个 DDE 服务器，如 Excel，在这种情况下不需要任何安全描述符，因为文件已在文件服务器的控制之下）。然而，大多数 DDE 或应用服务器的私有对象没有命名，它们需要另外的保护。Windows 也提供了一些 API 函数来帮助服务器保护自身。例如，CreatePrivateObjectSecurity 函数允许服务器附加安全描述符到它的任何私有对象上——如线程或其它进程。这些安全描述符能够防止除服务器外任何人访问其私有对象。

私有通信技术（PCT）

微软和 IETF 正携手开发一个名叫 PCT 的新的底层协议，和安全套接层（SSL）一样，PCT 是用来防止黑客窃听服务器和客户间的通信的，它通过加密、身份认证和数字签名技术来实现通信的安全性。由于可以使用 SSL，所以客户身份认证可以根据需要进行选择。

Web 链接 如果想了解有关 PCT 当前的进展情况，请访问 http://www.lne.com/ericm/pct.html，这个文档中包括了 PCT 的第二个版本的当前草案，另外，通过访问 http://www.w3.org/security/，你也可以参考一下 W3C 的 Web 站点上有关当前 Internet 安全技术的详细情况，其中包括 PCT 和 SSL。

PCT 假设你有一个可靠的传输协议，它能取代类似 TCP 的协议。有些人把 TCP/IP 看成是一种唯一的协议，但这是错误的，TCP 只是这些协议中的传输部分，而 IP 才是数据传输部分。IP 不支持任何形式的数据加密，所以，当你使用 TCP/IP 时,你的数据对任何想看它的人都是开放的。使用如 TCP/PCT 或 TCP/SSL 协议将使你的通信安全化。PCT 的第一个版本修正了在 SSL 中出现的一些问题，现陈述如下：

简化了的消息和记录结构。如果你没有使用客户身份验证，那么会话的重新连接在每个地址上需要一个唯一的消息；即便使用客户身份验证，重新连接在每个地址上也只需要两个消息。

扩展的加密算法。PCT 提供了比 SSL 更丰富的算法，这意味着它能支持更广阔的协议特性，而且这些协议特性的交涉是相互独立的。例如，公共特性包括加密类型、服务验证类型、散列函数类型和密钥字交换类型。

改进的消息验证密钥。在 PCT 中，消息验证密钥和加密密钥是相互独立的，这意味着，即便加密密钥是短的或不存在的，只要消息能使用一个很长的密钥，也能保证传输的安全，导制这一特性的主要原因是：因为美国政府把管理安全传输的密钥限定为最长为 40 位。

修补了安全漏洞。PCT 在会话中的客户身份验证是基于加密算法的。这防止了某些人通过捕获客户的验证密钥，并切断原客户的会话，而使用窃取的密钥来建立重新连接。客户只有知道了密文和密钥，才能获得服务器的访问。

附加的提前检验区域。客户和服务器在最初的握手对话过程中，其通信

是透明的，这个附加的区域使得客户和服务器可以检测在这种通信中可能出现的任何损害。

注释 虽然 SSL 在版本 3 中也提供了提前检验区域，但黑客也能够通过将协议版本号修改成 2 来绕过它，因为版本 2 中没有提供这一功能，而版本 3 完全兼容版本 2，所以客户和服务器都不会注意到这个变化。

微软当前正在开发 PCT 的第二个版本，这个版本与第一个版本完全兼容，但提供第一个版本所不具有的几个重要特性。以下是这些特征的简单概述。

新的数据报记录类型 单独的记录作为“数据报”独立地发送，从本质上看，这意味着协议不会保证数据发送的次序或到达目的地址的次序，这要求客户将接收到的数据进行排序，然后检验它们是否全部到达。这种方法的最大好处是提高了传输速度。

可重新识别的记录类型 记录的头包含了一些信息，它告诉接收者将接收什么类型的记录。

连续记录 PCT 版本 1 中允许数据跨越多个记录，即便是记录头没有指明连续的任何形式，版本 2 的记录头中增加了连续区，它也允许协议信息跨越多个记录。

数据记录的中间处理 数据记录是封装的，它允许发送者对数据做一些中间处理，比如压缩。

数据报记录的独立解密 由于发送的数据报记录可能会通过一个不可靠的传输通道，所以这部分特性对安全通信是非常重要的。每个记录分别加密，使得接收者可以立即对数据报解密，而不管接收数据报是否打乱

了次序。

新的密钥管理记录类型 这个记录类型允许发送者在会话中临时改变加密或消息验证密钥。重要的是，这允许了 PCT 来传送预先加密的数据。

新的关闭连接密钥管理消息 这是一个特定的消息，它告诉其它参与者关闭连接。因为这个消息是加密的，所以黑客很难发送一条伪装的消息来关闭这些连接。

增强型消息验证 消息验证现在也包括了记录头。

改进的握手协议 客户和服务器端的验证都包括了更加丰富的选项，其中有密钥交换、签名公共密钥和证书。

新的私有验证特性 这个特性允许客户和服务器使用预先一致共享的私有密钥进行验证，而不是使用公共密钥。

既然我们对 PCT 有了一些基本的了解，那就让我们看看 PCT 是如何工作的。PCT 使用可变长度的记录作为通信的方法，每个记录包括一个记录头，它定义了记录内的信息类型。一共有两类信息：应用信息和协议信息。应用信息总是包含数据，它们可以用标准的 PCT 格式或数据报格式；协议信息包含了密钥管理、错误或握手信息。PCT 使用两种附加的层，记录总是使用一个连接传送。客户和服务器间一般只有一个连接，但没有任何理由不能有更多的连接，每个连接只是会话的一部分。再强调一次，虽然你一般只能看到客户和服务器间的一个会话，但你确实可以用多个会话。（多会话可能需要客户和服务器间建立多个物理连接）。

PCT 协议的连接由一个握手信号开始，这就是握手管理消息类型开始工作

的地方。客户和服务器交换多个信息，首先是连接会话密钥的握手，或者说是客户和服务器决定一个用于相互对话的保密字。此时，客户和服务器也彼此进行验证。如果彼此认为对方不可信时，便不会进行这种握手；一旦客户和服务器都认为对方可信时，它们便约定好了一个用于信息加密的主密钥。

HTTP 上的 Windows NT 认证

Windows NT 的认证一般不是编程者要考虑的问题，更重要的是网络设置时要考虑的问题。但作为程序员，你确实应该了解一下认证问题，特别是如果你计划使用 Internet 或 Intranet 工作时更是如此。Windows NT 现在支持两种基本类型的认证。（虽然没有任何东西阻止一些人使用其它方法。）

技巧 如果你想使用 Windows NT 的质问/回答（challenge/response）特性，你必须使用 Internet Explorer 2.0 或更高版本的浏览器，其它浏览器当前还不支持这种实施安全特性的方法。

第一种认证的方法是 Windows NT 的质问/回答。这种认证方法依赖于服务器和客户密钥的通信，而没有用户输入的任何形式。在客户初始登录到系统时，服务器要求客户输入用户名和保密字，客户提供一个特殊加密的用户名和保密字，他还必须提供一个域名，因为客户必须在服务器的域内或在一个服务器所信任的域内。由于 Windows NT 的质问/回答在客户和服务器间自动使用加密通信，所以它比服务器提供的基本认证更加安全。

注 Windows NT 提供的两种形式的认证是基本认证和质问/回答，使用质问

/回答认证是最安全的。

Web 链接 有时，你顺手便可阅读到有关 Windows NT 针对 Internet 安全性方面的信息。其中一个信息源是 Rick 的 Windows NT 信息中心即 http://rick.wzl.rwth-aachen.de/cgi-bin/isindex3.cmd，这个 Web 站点涉及的一些内容，是你在微软的用户手册中所看不到的。例如，你将学会怎样使用 Windows NT 的各类第三方产品。在微软的方案中没有预先考虑到时，有些知识很值得学习。

那么客户是如何知道需要发送用户名和保密字到服务器的呢？服务器要求这些信息作为头的一部分，实际上，服务器发送了一个错误信息（401 Access Denied 即访问拒绝），它告诉用户需要一个安全访问。浏览器中实际上没有包括从服务器来的所有信息，懂得这一点是很重要的，因为你在浏览器中看到的只是服务器想要你看到的信息，它已经过浏览器筛去了头信息。例如，服务器一般必须告诉浏览器它正接收什么类型的信息，这样如果需要的话，浏览器便能够激活一个帮助。知道有这些头信息，对你了解在客户和服务器间的动态数据传输是非常重要的。

警告 如果你在你的站点上选择基本认证方式来实施安全性，你必须使用 SSL 来确保用户名和保密字传输的安全性。否则这些信息很容易被人破解，并利用它获得你的站点的访问。

第二种方法依赖于诸如 SSL 的数字签名技术。实质上，Windows NT 将需要一个从客户机来的数字认证，客户也需要 Windows NT 的数字认证。这些数

字认证可以从第三方厂家（如 VeriSign）获得，它能够保证双方的一致性。我们将会在本章的另一节中看到数字签名技术的确切过程。你现在应该知道的是，数字签名是一种安全手段，它能够标识试图访问你机器的另一方。

当使用 SSL 时，实质上是 Windows NT 服务器需要客户的标识。Windows NT 所获取的是由第三方厂家（如 VeriSign）发布的认证和随同的一个公共的密钥。SSL 在进行认证时，一共有以下六个步骤：

1. 客户发送给 Windows NT 一个没有加密的随机消息并伴随一个它的 VeriSign 发布的认证（它包含了客户的公共密钥），VeriSign 发布的认证是使用 VeriSign 的公共密钥加密的。因为每人都有 VeriSign 的公共密钥，所以 Windows NT 能够将整个认证解开并准确地检查它；同样，没有人能够伪造一个他们自己的公共密钥——他们必须从 VeriSign 那里获得。

2. 一旦 Windows NT 确认它从客户那里接收到了一个有效的证书和公共的密钥，它将告诉客户发送一个它最开始发送的消息的加密版本。

3. 客户计算机计算原始随机消息的摘要，然后使用公共密钥对它进行加密。

4. Windows NT 使用客户的公共密钥对这个摘要进行解密。

5. Windows NT 将解开的摘要和客户原始发送的随机消息进行对比。

6. 如果两个摘要匹配，这个客户便认证成功。

在这里，你可能会关心与管理员有关的问题，虽然 Windows NT 自身提供了这两种内建的方法来控制服务器的访问，但它并不阻止你创建 ISAPI 过滤器来扩编或替代这种标准的安全机制。我们在第 13 章看到了一些非常基本的过滤器，但你也可以干脆使用其它的方法。你所应该做的就是监视那些由服务器产

生的安全事件，这些事件中最关键的是客户的认证请求。在监视客户认证请求过程中，允许你增加任何必要的措施来确认谁在访问你的站点。

注　ISAPI 过滤器允许你改变 Web 服务器认证用户的方法。

使用数字签名

要精通数字签名的技术是比较困难的，虽然可以做到，但很少有人愿意去尝试。你首先必须懂得的是，数字签名也可以看成是授予证书。你可以把它想象成驾驶执照，因为它们之间有类似的功能。数字签名能标识一些 Internet 对象、以及创建它的人和创建的时间，也能潜在地提供很多其它信息，如果这个对象正好是客户或服务器，数字签名将显示对象的当前拥有者。和驾驶执照一样，数字签名也有过期的时候——它迫使提供商不断地实现它们的承诺，这些过期的数据也使得黑客在篡谋窃取证书上浪费许多无用的时间（因为每个证书都是一个分立的条款，所以黑客只学会如何窃取其中之一是没什么用的）。使用数字签名将帮助人们保持诚实，因为它迫使每个人通过一个中心核查点。在 Internet 上，使用数字签名做期货交易还避免了一大堆的问题：它不依赖于某个人来维护机器的安全性，现在你可直接对那些获得访问的人进行投资（这也意味着要对用户做某些级别的训练，让他们真正学会如何去使用这个特性）。

注　虽然代码符号处理（将在这一节的下载 Internet 部件部分进行讲述）现在还不是数字签名，但最终也将合并到数字签名中来。

实施数字签名是非常直接而简单的，尤其是对客户端。在大多数情况下，商家提供了标准的证书，它能被所有浏览器或服务器所识别，但在授予证书的

实际过程中，它们使用的方式却有不同。例如，在图 14.2 中的 Web 页中便显示了其中的一个潜在的问题，如果你想要获得一个客户的证书，你将选择哪个等级呢？（因为 Class 1 Digital ID 证书 的价格便宜，所以许多人将会首选它）。

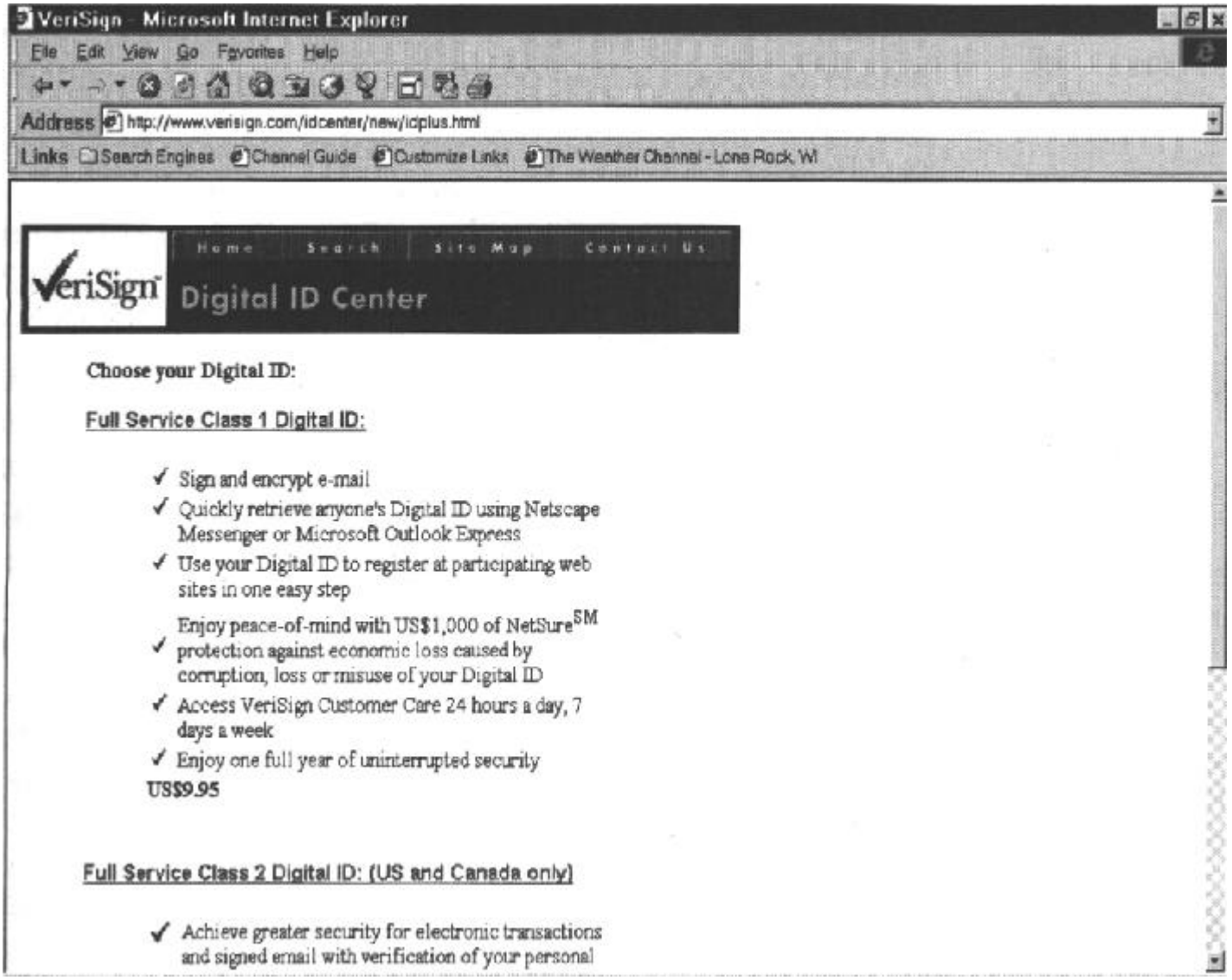


图 14.2　客户端的认证将是未来 Internet 安全性的一个重要部分，但到底选择哪种认证并不容易

从实施的立场来看，你获得了 Class 1 还是 Class 2 证书对你的安全实施并不重要，即便是厂家，也不真正重视它。然而为了兼容性，你可能仍然想坚持用 VeriSign（其实不然）。事实上，不同厂家使用数字签名的方法是非常相似的，因为它们的界面都是由 Web 服务器或浏览器所定义的，你将要做的只是接受一个应用程序，当厂家核查了你的身份后，你将会在 e-mail 中获得一个 PIN 或其它方式的证明，关键是要按照浏览器或 Web 服务器的提示安装这个证书。例如 VeriSign 公司，你进入它的 Web 站点并输入你的 PIN 后，浏览器将自动关注下载和安装的细节。下述段落对 VeriSign 的证书等级作了一个概述。

Class 1 证书，它在一个资源库内提供用户唯一的名字和地址，VeriSign（或你选择的任何其它厂家）将能够核查这个人的姓名和地址。回发 e-mail 是 VeriSign 用来发授证书的唯一手段，要接收这个证书，你必须拥有一个 e-mail 地址，这种方式使得黑客很难伪造这个证书。这种级别的证书每年需花费 9.95 美元（你也可以获得一个 Class 1 证书的免费测试版）。

Class 2 证书，要获得 Class 2 证书，必须由第三方认证你的身份（目前只对美国和加拿大人授予 Class 2证书）。Class 2和 Class 1的最大区别是 VerSign 公司实际通过 EquiFax 公司提供的用户数据库来检测你的信息。你也通过一个硬件签名处理，它需要多个密钥。这种级别的证书每年需花费 19.95 美元。

如果你想获得一个证书并希望它能和多种软件协同工作，那么你不能只看它对浏览器的兼容性，因为数字签名技术有很多的标准规范（请参看表 14.1），

你将发现大部分的数字签名认证厂商已接近这些标准了。

如果你已经有了一个数字签名的证书，并分配给了你的 ActiveX 控件、Web 服务器或浏览器（或以上所有的部件），那么你怎样鉴别其他人有没有证书呢？在你访问不安全的站点时，你总看到一些警告对话框，而访问一个有数字签名证书的站点也与之类似，你将看到一个如下图所示的数字签名证书。



注意，这个对话框还给了你一些优化选项。首先，你要检测证书的有效性，例如，检测它显示的厂商是否是你所期望的那一家；检测其日期来判断是否过期。其次，如果你想要到这个厂商那里去，你应该有能力通过其审计。在证书下的第一个检测框总是允许你在列表中增加一些由 WinVerifyTrust 检测过的指定的公司（我们在第 8 章讲述下载过程中提到过 WinVerifyTrust），如果你选择这个厂商，则每次需要在这个指定的公司下载东西时系统将不再提问，这对访问某些站点来说可能有点冒险，但也许没比这更好方法了，它完全决定于你对这个公司的信任度；第二个检测框允许你对所有授予了指定证书代理权的商家进行访问，如 VeriSign，如果你的认证过程很舒适，你可以不选中这个检测框。

你也许正为要对 ActiveX 控件所做的这些事而感到惊讶。用户和 Web 的主人一样，如果你决定使你的站点公共化，你将发现自己在某些地方需要这些证书。因为人们不愿下载一个未经数字化标识或看不到厂商证书的控件。取得证书的过程应该与我们刚看到的浏览器用户获取证书的过程没有太大的区别。你现在不必立即拥有一个证书来标识你的控件，但在不久的将来，它是必不可少的。我们将在本章有关下载 Internet 部件的部分，介绍标识 ActiveX 控件的实际过程。

理解加密 API

到此为止，防止某些人偷阅你的数据一直是本章所有内容的主题。我们已经学习了一些安全协议和技术，它们能帮助你将黑客拒之门外。然而，只要有

充分的时间，任何一把坚固锁都可能被人撬开。事实上，这就是为何要使用 128 位的密钥代替 40 位密钥的原因。由于解开 128 位的密码“锁”需要太多的时间（至少现在是这样），所以包含在这个记录内的数据对黑客几乎没有吸引力。黑客当然也能打开这把“锁”，但非常困难。

保护层

增加保护层是针对黑客的另一种利器。对数据进行多种级别的加密并加上“防盗门”，黑客要获得数据必须通过这些“门”。只要在你的数据和黑客之间放置足够多的“门”，黑客便很难攻破它。

微软的加密 API（或加密应用程序接口）增加了保护层类。它意味着你又可以对敏感数据增加一个保护层了。这里支持的加密技术虽然并非坚不可摧，但黑客要解开它读取你的数据，确实需要花费更多的时间。使用加密 API 函数将帮助你更好地保护服务器和客户间的数据传输。

Web 链接 对于其它涉及 Internet 的东西来说，加密 API（CryptoAPI）是全新的东西，并且还在不断地发展之中。在撰写本书时，加密 API 只是作为 ActiveX SDK 的一部分来叙述的，你应该通过 Internet 下载它的一个详细说明。虽然本书在这一部分提供了最好的叙述，但你还是应该访问一下加密 API 站点 http://www.microsoft.com/security/tech/misf6.htm，以获得最新信息。

加密 API 还有其它一些用途。因为加密 API 是作为一个通用工具而设计的，

所以它允许在任何环境下对数据进行加密，而不只限于 Web 服务器。例如，你可以创建一个应用程序将所有存储数据使用相同的加密格式，这些数据可以是存储在本地的、通过 modem 传输的、上载到一个 Web 站点的，或是通过邮件发送的。这将给你的公司带来很多的好处：对敏感数据进行随时加密，使得黑客即便闯进了你的系统，也很难破解你的数据。然而，使用加密技术，也会给你的用户带来一些不便，可以预料，只要加密操作变得更加方便和自动化，你的用户肯定会喜欢上它的。让攻击者的攻击变得困难、摸不着头绪，或者简单地要他们花费更长的攻击时间都是吸引用户的最佳途径。这样麻烦不断的用户就会确信应该在你的系统中保护他们的数据。

注释 Windows NT 4.0 内建了加密 API 1.0（CryptoAPI 1.0）支持。当你安装完服务器修补包 3 后，其支持的等级将升级至 2.0。这两个版本的最大区别是：加密 API 2.0 支持核心加密，它允许开发者在应用程序内纳入加密，而且加密 API 2.0 还支持 X.509 证书、ASN.1 编码、PKCS#7 和 PKCS#10 压缩。你将发现，Visual C++需要有 WINCRYPT.H 和 CRYPT32.DLL 文件，才能提供 CryptoAPI 2.0 的支持。

CryptoAPI 是模块化的，这并不足以为奇，因为在过去的几年里，微软将它的操作设计转移到了模块化的方式上。我们可以拿 CryptoAPI 与 Windows 下的 GDI（图形设备接口）API 相比较，任何厂商都能够增加新的设备驱动程序来告诉 GDI 如何和指定的显示卡工作。CryptoAPI 也是如此，它采用了加密服务提供者（CSP）的思想。虽然 Windows 本身已有微软提供的 CSP，然而，如

果它的加密特性还不能满足你的需求时，你也可以自己设计一个新的 CSP 或从别的公司购买一个。CSP 的安装过程就像是增加一个设备驱动程序一样。事实上，这种设备驱动程序的方式使得在你的机器上可以混合加密硬件和加密软件，这是一个真正的提高点，因为你通常是将硬件和软件作为分立的实体来安装的。

Web 链接 微软已经为 CryptoAPI 开发了一个 DDK（设备驱动程序开发工具），Cryptographic Service Provider Developer's Kit（CSPDK）。你可以通过 http://www.micorsoft.com/devnews/novdec96/crypto5_6.htm 获取更多的信息。

加密一个文件

为了显示一下 CryptoAPI 提供的机能，下面举一个加密文件的例子。加密文件是相当简单的，微软直接依照八个步骤处理（其中六步含有特定的加密函数）。最开始，你象在其它编程中一样，打开一个源文件和目标文件（例子中的这部分代码用C语言编写，你也很容易在其它语言中实现），在你拥有一个有效的文件句柄后，你必须像下边的例子一样获得一个 CSP 句柄：

```
//给默认提供者提取句柄
if (!CryptAcquireContext(&hprov,NULL,NULL,PROV_RSA_FULL,0))
{
   MessageBox("Error during CryptAcquireContext",NULL,
 MB_OK |MB_ICONEXCLAMATION);
```

```
    PostQuitMessage(1);
}
```

注释 在 WINCRYPT.H 文件中，你能看到与加密有关的函数、结构和定义的完整列表。

CryptAcquireContext()能够接收五个参数，但在这个例子中只有两个是重要的，第一个参数保存 CSP 的句柄，第二个参数指定你要查找的 CSP 类型。每个 CSP 都有一个名字和类型。例如，当前随 Windows 一起发布的名叫 Microsoft Base Cryptographic Provider v1.0 的类型是 PROV_RSA_FULL。每个提供商的名字是唯一的，而类型却不是唯一的。第二个参数包含了密钥的容器名，如果你在这里输入一个值，Windows 将寻找一个指定的密钥容器，厂商可以在硬件里、注册表中或硬盘里保存一个密钥容器，所以你一般不会知道密钥容器的名字，如果使用 NULL 值（如例子中所示）将告诉 Windows 返回默认的密钥容器；第三个参数包含了 CSP 的名字，你可以通过使用 CryptGetProvParam()函数来取回这个值，填写 NULL 值将返回默认的 CSP；最后的参数包含了一个或多个标志，如果 CSP 没有提供指定的标志值，你可将这个参数设置为 0（微软只为管理方面提供了一些默认的标志值）。

下一步可以是使用一个随机密钥或由口令产生的密钥来加密文件。随机密钥方式在 ActiveX 控件中使用的可能最为频繁，所以在此加以描述，口令方式也与之类似，而实际需要的步骤比随机密钥更少。

```
//创建一个会话的随机密钥
```

```
if(!CryptGenKey(hProv,ENCRYPT_ALGORITHM,CRYPT_EXPORTABLE,&hKEY))
{
    MessageBox("Error during CryptGenKey",NULL,MB_OK |
MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}
```

CryptGenKey()函数提供给你一个唯一的密钥。注意第一个参数是 CSP 句柄，第二个参数包含了加密算法名。微软提供了两种加密算法：CALG_RC2（块密码）和 CALG_RC4（流密码）。算法名会随着厂商的不同而变化，所以知道用户打算使用哪家厂商的 CSP 是很重要的。第三个参数包含了标志，一共有三个默认的标志值，CRYPT_EXPORTABLE 告诉 Windows 能够输出一个随机的密钥到一个 blob（我们将在随后看到它）；CRYPT_CREATE_SALT 告诉 Windows 给随机密钥种子值使用一些其它大于 0 的值；最后，CRYPT_USER_PROTECTED 告诉 Windows 在某个动作发生时提醒用户。最后一个参数是存储返回随机密钥的一个容器。

在获取一个证书的过程中，用户将接收到一个公共密钥，CSP 将这个密钥保存在一个中心位置——用它标识用户。为了保证数据传输的绝对安全，在加密过程中，获取用户公共密钥的一份拷贝来使用是很重要的，这就是下一步所要做的工作。

```
//获得用于交换密钥的公共密钥句柄
if(!CryptGetUserKey(hProv,AT_KEYEXCHANGE,&hXchgkey))
```

```
{
    MessageBox("ErrordwringCryptGetUserKey",NULL,MB_OK |
MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}
```

我们提供给 CryptGetUserKey()**的第一个参数是** CSP **的句柄，第二个参数是表示从提供者那里获取什么类型的密钥。每个** CSP **将支持两种密钥：** AT_KEYEXCHANGE**（这个交换的公共密钥是为应用程序中的个人认证而提供的）或** AT_SIGNATURE**（与随后的** ActiveX **控件获取数字签名密钥一样），第三个参数提供存储返回密钥的空间。**

我们现在进入了一个有趣的部分。我们获取随机密钥和用户的公共密钥，并将其混合使用，这意味着，即便用户的密钥在某些方面被破译了，数据还仍有另一种方式的加密，它迫使黑客又得重头开始破译。随机因素使得加密技术至少在安全上超过了其它方式的加密。

```
//决定键 blob 的大小并分配内存
if(!CryptExportKey(hKey,hXchgKey,SIMPLEBLOB,0,NULL,&dwKeyBlobLen))
{
    MessageBox("Error computing blob size",NULL,MB_OK |
MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}
```

```
if ((pbKeyBlob = malloc((dwKeyBlobLen)) == NULL)
{
    MessageBox("Error allocating memory for blob ",NULL,MB_OK |
MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}

//将会话的密钥输出到一个简单的密钥 blob 中
if(!CryptExportKey(hKey,hXchgKey,SIMPLEBLOB,0,pbKeyBlob,&dwKeyBlobLen))
{
    MessageBox("Error during CryptExportKey",NULL,MB_OK |
MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}
```

注意输出 Blob **密钥（用户的公共密钥和随机密钥的组合）的过程，实际上有三步处理，我们调用了** CryptExportKey()**函数两次，头两个参数值是我们创建的密钥：随机密钥和用户密钥；第三个参数告诉** Windows **创建什么类型的** blob**，大部分** CSP **支持两种值：**SIMPLEBLOB **或** PUBLICKEYBLOB**；第四个参数是一些标志，除非** CSP **为别的目的而使用它，否则这个参数总设置为** 0**；第五个参数是指向存储** blob **的缓冲区的指针，如果你在调用** CryptExportKey()**时将它**

设置为 NULL，它将简单返回缓冲区的大小，这个大小在第六个参数中将用到，这就是我们在第一次调用它的目的；第六个参数一般包含了你实际调用 CryptExportKey()创建 blob 时所需缓冲区的大小。（这里的源代码也包含了一个内存分配函数的调用）

既然我们已经获取了一个 blob，那么现在该关注一下后续工作了，你马上要做的工作是销毁用户的公共密钥，因为黑客可以从你的内存中搜寻到你的密钥并破坏你的代码，所以从内存中将密钥删除，无论从内存的角度还是从编码练习的角度来看，这都是一个很好的策略，它对维护你的安全系统来说，是必不可少的。下面的代码是调用 CryptDestroyKey()函数来删除公共密钥。

```
//释放键交换密钥的句柄
CryptDestroyKey(hXchgKey0;
hXchgKey = 0;
```

在这时，你可以将 blob 数据保存到磁盘或硬盘内的一个文件中。blob 实际上组织了一个头，接收的机器将使用它对文件进行解密。然而，一个没有数据的头并不好，这就是我们下一步所关注的，你要做的第一件事就是定义数据块的大小，如果你使用块加密方式，CSP 将提供块大小，块加密一般需要你在文件的尾部增加一个附加的空白块。当使用流加密时，你可以定义任何大小的加密块（尽管微软推荐使用 1,000 字节的块大小）。现在你将数据一块一块读取到缓存，并使用以下代码对它进行加密，然后将数据块写入目标文件中。

```
//对数据进行加密
if (!CryptEncrypt(hKey,0,eof,0,pbBuffer,&dwCount,dwBufferLen))
```

```
{
    MessageBox("Error during
CryptEncrypt",NULL,MB_OK|MB_ICONEXCLAMATION);
    PostQuitMessage(1);
}
```

你可以设计一个循环来不断地处理数据，直到读完数据块为止。确认你填充了所有不满的块，CryptoAPI **甚至将块大小作为加密处理的一部分。**

14.3 蛮荒不化的互联网

一些人把 Internet **看作是计算机的现代野蛮西部，因为在** Internet **上实施的标准实在太少，几乎没有什么安全手段。虽然在** Internet **上出售商品的人正以指数方式增长，但不幸的是，蛮荒不化的** Internet **环境使许多慎重的行业，如银行和大型零售商场都不敢问津。造成这种情况的原因很简单：它们不愿意承受可能出现安全漏洞的风险。所以，就在许多新兴的公司，如** CD-Now **正高兴地迎接** Internet **的同时，另一些需要** Internet **的老牌公司却对它持观望的态度。**

虽然 Internet **技术在不断成长革新，但它还是招致了“现代野蛮西部”的“雅称”。**Internet **是内部技术、新技术和老技术的结合体，然而，这正是威胁** Internet **的最大原因。例如，当前版本的** IP **协议（互联网协议）不支持任何方式的加密数据（至少在协议级没有支持），问题是，如果人们想通过加密的形式在** Internet **上传送数据，将没有工具来确认其安全性是完全可靠的。新版本的** IP **将解决这**

个问题，但现在还尚未出台，随着时代的发展，Internet 必定会在这方面取得成功。现在虽然有那么多这样或那样的升级承诺，但是很少有实现的。对于某些人来说，在 Internet 上“冲浪”是非常有趣的，因为它是那么自由自在、无拘无束。

Web 链接 通过浏览 http://www.genome.wi.mit.edu/www/faqs/www-security-faq.html，你可以看到在运行 Web 服务器时所要担负的一些安全风险。它包括了使用某种浏览器或服务器时所冒的风险，其中包含了当前的 Windows NT、Unix 和 Macintosh 服务器，最后它倾向于使用 Unix 服务器。该站点也给出了许多通用问题的答案，比如，如何在你的严格监视和保护用户的隐私之间找到一个平衡，它甚至还告诉你，文档文件是如何随着 Internet 的发展而变化的。

就像野蛮的美国西部最终发展成了现代文明国度一样，Internet 也将最终走向辉煌。类似 ActiveX 控件这样的技术将使 Internet 最终变成一个安全的商务场所，这些技术也将把 Internet 变得更加友好；就像汽车和高速公路把美国西部变得更加容易旅行一样，这些技术也将加快人们从一个地方到另一个地方的交流速度。

我们必须要解决的一个难题就是安全问题。只有在一个安全的环境内，人们才能够安心高效地工作，否则，人们将在与敌人的死死纠缠中花费太多的时间和精力。下述各节将讨论隐藏在 Internet 安全问题背后的相关课题。你在应用程序内处理安全问题之前，应该先弄清楚有什么样的安全问题，如果你现在已经明确了所有安全问题，并想实施安全策略时，请看看讲述 Internet 安全标准的

那一节，否则，如果你想把 Internet 作为一个工具应用于你的公司，你应该先花点时间来思考一下将要面临的一些安全问题。

从商业角度看 Internet

就像前面提到的，许多人为毫无限制的 Internet 感到不安，你确实可以在 Internet 上看到几乎任何东西，因为没人对它进行规划（美国政府最近对 Internet 在线服务的规定并没有奏效）。如果你是一个编程人员，Internet 这个无拘无束的环境也许还有某种魅力，但你不是在为自己编程，你必须考虑到商业人员的需求，商业人员当然不希望花了钱却买回了不安，四面漏风的工作环境对他们来说就像是一场噩梦，他们希望每件事都井然有序，有条不紊，最重要的是要安全可靠。

急速膨胀的 Internet 并没有对商业用户注入多少信心。在一些商业用户的眼里，新的和未经考验的技术只会在茫茫的等待中发生。考虑一下 ActiveX 控件的潜在得失，它确实加快了商业用户的办事速度，但还是有一些安全因素值得担忧，因为任何一个具有少量编程经验的人就可以创建一个潜伏破坏性的控件，并把公司的一切搞得一团糟。

最近的商业调查报告显示，大公司在 Internet 上变得审慎了，他们采用审慎方案的一个最大原因是缺乏安全感，这些公司希望能证实技术的可靠性，并保证它们的数据在任何通信中都很安全，建立这种值得信赖的 Internet 的一种方法是在内部编写优良的应用程序，也就是说，创建 ActiveX 控件。

注释 商业用户需要处理的主题很多，它们一般与编程人员无关，例如，是否允许职员访问 Internet 以及允许进行什么层次的访问？这些当然是个热门的话题，但我们在这里不做全部的描述，我们只是讲述一些与编程人员相关的主题。

那么你该如何处理这些安全问题呢？你可以与公司的主管交谈，如果你发现了他们最担心的只是你编写的应用程序，那么你应该说服他们，作为应用的一部分，安全也是非常重要的。例如，我们可以使用一些方法来访问 Windows NT 服务器内部提供的安全性，这种方法的缺点是访问安全特性将需要许多附加的代码，使得应用程序的运行变慢，并使用户也更辛苦，你也应该在程序的设计阶段告诉主管对其所进行的权衡（应用程序最终将在 Web 站点上应用）。

注 要实现公司各阶层间的通信，必须建立一个安全的 Internet 或 Intranet 环境。

定义要保护的对象：数据

那么，我们需要保护的到底是什么东西呢？你也许会得到许多答案，但它们实质上都是指向一个东西，所有说的和做的都是为了保护数据的安全。数据所牵连的影响甚至超过了硬件和软件，硬盘丢失了，你还可以替换它；软件配置破坏了，你还可以重新安装它，可是公司的财经周报丢了，你就得重新处理它了。数据是计算机环境中最精确的部分，因为它一旦丢失，便很难再修复。

保护数据的安全是所有人在使用中最应该关心的事。很多网络管理员在照

顾安全事务上花费的时间最多，在大多数情况下，安全性关系到了谁做了访问、访问了什么数据以及访问的原因等。随便翻翻任何一家商业出版物，你都能看到一篇又一篇的文章讲述数据的安全性。我们每时每刻都在用计算机处理数据，这些数据对你公司的重要性是毋庸置疑的，当你创建一个 Active 控件，并连接到 HTML 页面上，此时你最大的安全顾虑就是你将要提供的数据。

数据安全涉及到了访问的问题，不管是访问本地机器还是访问网络，有一个目标是相同的，那就是防止非授权的访问。在本地机器建立一个安全的环境是相当容易的，除了有各种软件技术外，许多计算机还允许 BIOS 层次的口令保护，如果这还不够，你还能依靠物理的安全保护来保证别人访问不了你的系统。然而，要在网络上实施这些安全性就比较困难了，但你还是不用花太多的努力就可以做到，流行的网络操作系统产品如 Novell 的 NetWare 和微软的 Windows NT，都提供了各种安全手段来保护本地数据（除非有人在网络的物理电缆上安装上了网络监听器）。在广域网上（WAN）实施安全性简直就是做恶梦，尤其是当你还有许多拨号上网的机器，不管连接的距离如何，由于它们使得软件和硬件的项目变得更丰富了，所以某些人便更容易偷偷闯进你的网络。有些人甚至断言，在 Internet 上实施安全性是不可能的事。不只是广域网带来了这个重大的问题，Internet 自身提供的公共访问也带来了一些问题。任何人，只要看过最近的商业杂志便知道那些软件公司是如何倾尽全力地来修补他们的产品中的安全漏洞的。

建立某些形式的保护

即便你现在尽量忽视 Internet 方面的安全问题，数据保护还是有许多其它方面的主题。例如，你将提供什么类型的数据保护？让我们暂时从非 Internet 的角度来探讨一下，在这里，你能够从硬件或软件的角度来看待安全保护。一些公司为了保护那些需要特殊照顾的数据，他们把工作站从其它项目中分离出来，与网络的物理分离意味着，任何人如果想要访问机器内部的数据，必须实际地使用这个工作站，必要的话，你还可以通过使用加锁来防止别人访问它。有的公司还通过使用数据加密或执行安全审计来解决这个问题，这些都是用软件实现的安全策略。

不幸的是，在 Internet 上，你不可能把自己的机器锁在房间内来达到保护数据的目的，这意味着，随着 Internet 的发展，过去使用硬件方式保护数据的方法将会被淘汰。硬件级保护的失去增加了编程人员的负担。作为一个编程人员，你必须负责在应用软件中增加一些安全手段（包括一些产品如 ActiveX 控件，ISAPI filters/extensions）。幸运的是，你能够参考许多新的安全方法，例如，文件加密已经是一个实用的保护技术。无论在广域网上还是 Internet 站点上，加密文档的安全特性都一样，只是允许访问的人数不一样。

我们已经讲述了工作站的一些安全问题，让我们再来关注一下服务器。从 Internet 的角度来看，有些公司把自己的站点和网络从物理上完全脱离开来，（这限制了你将这个站点用于公司内部需求的方法）。使用一个分离的 Web 服务器是保护你网络安全性的一种物理方法，没有人能够通过这个 Web 服务器来窥探你网络内部的数据。防火墙是一种软件的保护方式，这种方式的登录过程，实

质上在局域网上已使用过多年了。

你当前使用的或以后将要实施的计划都有一些实际的极限，其中，你最应该考虑的是 Internet 访问的发展将越来越灵活。当然，你可以关闭你的站点来防止任何人来访问它，但你此时又能做什么呢？采取顽固和粗糙的策略来实施你的安全性可能不会有太大的效果。为了让人在 Web 站点上学习加强安全性的新方法，而不分配给他们访问的级别，将是你面临的一个挑战，从编程者的角度来看，这意味着你应该学习安装在网络上的硬件和软件的应用能力，已避免陷入 Internet 的安全顾虑中。

实施方案

要创建一个安全网络环境是一个庞大的项目，即便是系统地陈述一个基本的安全方案，你也要花费许多时间和精力，而要具体实施它则需要更长的时间，尤其是为了减少对工作环境的冲击，而使用阶段性的实施方案时，时间就更长了。即便你制定了一个非常安全的计划并尽己所能地考虑到了所有的意外，你也不能完全确信已经面面俱到了。当最后有人闯进了你的安全系统并破坏了一些数据时，你便只能打道回府了。

一旦考虑到所有这些潜在的问题时，你不难得出为什么网络管理员要经常地对用户的权限进行分配，哪怕他是在做一件无关痛痒的事。例如，一些网络管理员强烈地反对公司进行任何形式的 Internet 访问；而另外一些网络管理员则一旦证实有 Internet 访问，便要像“老大哥”一样监视用户的活动情况。

不管建设什么形式的 Web 站点，你面临的将不止是网络管理和数据保护的

问题，考虑用户的提问也很重要。如果有人告诉你连接是安全的，而你发现它并非如此，你又怎会喜欢它呢？许多信用卡用户真不愿意这样冒险，从计算机到 Web 站点的一个不安全的连接可能意味着信用卡内数千元的损失。

在实施一个安全方案之前，你应该已经完成了这些工作：确认你创建的应用程序能够长时间保证客户和服务器间数据交换的安全；如果你想把你的 Web 站点作成防弹（bulletproof）形式，你应该考虑到所有使用 Internet 人的需求，以及可能产生的冲突。当一些没有考虑到的事情发生、并且确实有人闯进了你的网络时，你便能够立即制定一个安全计划来修补这些网络安全漏洞。

14.4 确保 Internet 下载代码的安全

许多公司正承诺把它们的应用软件当作一系列 ActiveX 控件上载到 Internet 上，例如，Lotus 计划将一系列 Notes_specific 阅读器以 ActiveX 控件方式迁移至 Internet 上，Quarterdeck 公司也计划把它们的整个应用系统程序转换成 ActiveX 控件形式并上载到 Internet 上。你很快就能够直接通过浏览器查看各种类型的数据，在某些情况下，你还可以直接对它进行编辑。（要得到它们成熟的产品，我们也许还得等待一段时间）。

如果商家没有计划对下载的代码进行检验、用户在不能确认代码是否受到过病毒的侵害时，就盲目将其保存，将会带来很大的危害。我们已经在第 8 章中阐述了 Windows Verify Trust API（用于连接 Windows Trust Provider Service）背后的一些技术，在这一节中，我们将关注这些 API，你将对这些技术的运作

方式有一个总体的了解。

在这一节中，我们也将关注一下 Windows Software Publishing Trust Provider，它实际上是Windows Verify Trust API的一个附加系统，它用来检验下载的软件部件是否值得信赖，其检验的方法有多种，其中包括检验本地的规则（如在浏览器上与安全相关的检查框）和检验文件自身的加密信息（如数字签名）。

最后，我们将关注对ActiveX控件签名的过程。将签名过的控件放置到Web服务器上，将使用户在下载它时，不会再提示“not trusted”（不能信赖）的信息。这部分也将让你学会如何为测试而清理自己的机器，你将学会如何卸载一个ActiveX控件，这样，你不必在每次测试控件的签名以及在你的Web站点下载其它特性时，都得重新清理机器。

使用Windows Verify Trust API

这个特殊的API在某些方面现在已经相当稳定，微软已经在IE3.x/4.x以及Windows自身中部分地实施了它。你阅读到的其它部分正处于不断的改变中，因为很多软件公司正在各自地研究检验下载软件的最好方法。

那么，Windows Verify Trust API的精确定义是什么？它是决定你是否能信任任何Windows对象的一种通用方法，这些对象可以是客户请求的服务、服务器请求的信息、下载的文档文件或甚至是ActiveX控件。这个API的最终形式是允许你检测任何对象的可信度。

就像Windows支持的大多数API一样，Windows Verify Trust API也是可以

扩展的，你可以增加新的特性来允许它执行一些扩展的检测。其中的一个扩展是 IE 3.x/4.x 带来的 Windows Software Publishing Trust Provider，我们将在下一节进行阐述。现在，你所要知道的一点是，Windows Verify Trust API 只是个通用的 API，随着更多人对它的使用，它可能需要更多的扩展。

Windows Verify Trust API 使用多种方法来检测文件的可信度。其中的一些方法尚存在争议，但有两种方法最为通用，即检测系统规则和校对伴随于对象的证书或数字签名。你将发现 Windows Verify Trust API 也依赖于外部的证书，例如，当前许多流行的 Internet 加密标准正使用公共密钥和私有密钥方式，公共密钥驻留在文件头上，私有密钥存在于用户的机器内。要破解一个加密文件，你必须要既有公共密钥又有私有密钥，因为只有用户拥有这个密钥，所以其他人将不能阅读这个文件。显然，还有一些比两把密钥更复杂的方法，有些简单的方案还增加了随机密钥，他与公共密钥相结合，使得黑客几乎没有可能破解它。

系统规则存在于许多地方，例如，浏览器的“信任”信息存储在配置文件或注册表中。系统管理员也可设置一些策略，这些设置可以在一个单独的用户注册文件中或在所有用户都可使用的通用注册文件中。策略所在的精确位置依赖于你使用的 Windows 版本，和是否打开了多用户配置（在 Windows 95/98 中），以及系统管理员实施的策略类型（系统的或单独的）。你也可以使用委托供应商（CPS）规则（准确的名称叫做“trust provider”，因为其中的规则来源可以是任何委托代理，而不止是 CPS），例如，你可以告诉浏览器，任何经过某个委托供应商认证的人都是可信的。作为受托检验规则，委托供应商提供了指定

的拥有者的列表，并定义了指定对象的类型和委托供应商的级别。用户的操作也能够影响 Windows Trust Verification API 的使用规则，例如，如果用户告诉系统，某个供应商是可信的，则这个信息便存储于注册表中，而 Windows 的受托检验服务每次在证书中看到这个商家的名字时，便不再对这个对象作深入的检测。

现在，你已经对这个 API 的功能做了一个概要的了解，让我们再关注一下这个 API 的自身。你将会对下面这个函数感兴趣：

```
WinVerifyTrust(HWND hwnd,DWORD dwTrustProvider,DWORD dwActionID,LPVOID
ActionData);
```

正如你所看到的，这个函数需要四个参数，大部分人应该牢记第一个参数，它是当前窗体的句柄，这个参数的目的是让 WinVerifyTrust()函数知道当前正有用户在做决定，例如，函数可能想询问你是否下载一个没有签名的文件。如果想要在不干扰用户的情况下检验一个对象的可信度，你可以简单使用 INVALID_HANDLE_VALUE 来代替窗体的句柄；如果你想要用户的桌面代替当前的应用程序，对任何交互作用都做出反映，你也可以将参数值设为 0。第二个参数定义了对谁来询问与信任相关的事情，Windows 认识两个默认值（尽管商家也能定义任何指定的值用来明确真正的委托供应商）：WIN_TRUST_PROVIDER_UNKNOWN（基于你想要执行的操作，找到一个代理商）或 WIN_TRUST_SOFTWARE_PUBLISHER（一个实际的软件发布者）。如果你选择 WIN_TRUST_PROVIDER 选项，Windows 将尽量找到一个包含你要执行操作的注册项。如果不能发现这样一个注册项，WinVerifyTrust()函数将

返回一个 TRUST_E_PROVIDER_UNKNOWN 的值。第三个参数指定了一个操作，它告诉代理供应商，你想要做的操作，因为每个代理供应商都不同，所以你将必须检查代理商提供的文档，最后一个参数的准确内容也依赖于你使用的代理商。在大多数情况下，你将至少需要告诉代理商你想要检验的数据，有些代理商也可能要求有关信任级别的信息或者信任裁决的内容。

WinVerifyTrust()函数在调用执行后，一般返回代理商的特定值；在某些情况下，它也返回四个标准值中的一个，你将注意到这四个标准值全是错误信息（Windows Verify Trust API 没有定义任何默认的成功信息），下面就是这四个值的解释。

TRUST_E_SUBJECT_NOT_TRUSTED　通常，代理商会提供给你更多的错误信息，这个返回值简单地提示：检验的对象没有通过你指定的检验操作，它不代表这个对象从根本上不值得信任，而只是不能被当前指定的操作信任。除非代理商支持某些类型的通用操作，或者你只需要对这个对象执行单一的操作，否则你将需要为每个操作分别调用WinVerifyTrust()函数。

TRUST_E_PROVIDER_UNKNOWN　就像前面提示的，Windows 基于当前的定义操作，不能找到指定的代理商时，便返回这个错误信息。

TRUST_E_ACTION_UNKNOWN　如果你要求的操作不被指定的代理商支持时，函数将返回这个错误值。WinVerifyTrust 使用注册项来检验有效操作，而不是真正与代理商进行交流，这意味着，即便代理商支持指定的操作，你也可能会因为注册被损坏而不能执行某个操作。

TRUST_E_SUBJECT_FORM_UNKNOWN 产生这个错误信息的原因是多方面的，多数情况下是因为参数不正确或参数中包含了不完整的信息。如果代理商不能找到你想要检验的对象，也将返回这个信息，幸运的话，代理商将提供与数据相关的更详细的信息，但如果代理商不能判断问题的来源时，你或许也只能得到这个值。

理解 Windows 软件发布信任供应商

Windows 软件发布信任供应商（Windows Software Publishing Trust Provider）附加于 Windows Trust Verification API（我们在前一节中已进行了阐述）之上，这个附加系统的主要目的是允许应用程序检验软件部件中是否包含数字签名或证书，其中的每个项目都将检验用户本地系统中的文档，并判断它们是否是由委托的供应商作为权威的软件发行的。对于 Windows Trust Verification API，这个 API 使用丰富的技术和信息资源来判断指定的文档是否值得信赖。

你将发现 Windows 软件发布信任供应商也使用 WinVerifyTrust()函数，在前几节中我们讨论过这个函数，但这里的使用略有一些不同，首先，你总应使用 WIN_TRUST_SOFTWARE_PUBLISHER 委托供应商，除非委托供应商支持别的值，如果使用 WIN_TRUST_PROVIDER_UNKNOWN 委托供应商，Windows 将简单选择默认的委托供应商，Windows 也定义了两种操作（你可以提供其它可供选择的委托供应商）：WIN_SPUB_ACTION_TRUSTED_PUBLISHER（检验文档的供应商是否在信任列表中）以及

WIN_SPUB_ACTION_PUBLISHED_SOFTWARE（检验文档自身是否有正确的鉴定证书）。当前版本的 Windows 软件发布信任供应商还不支持 WIN_SPUB_ACTION_ TRUSTED_PUBLISHER 操作，如果选择 WIN_SPUB_ACTION _PUBLISHED_SOFTWARE 操作，WinVerifyTrust()也将期待 WIN_TRUST_ACTDATA_ SUBJECT_ONLY 数据结构，该结构如下所示：

```
typedef LPVOID WIN_TRUST_SUBJECT
typedef struct WIN_TRUST_ACTDATA_SUBJECT_ONLY
{
    DWORD  dwSubjectType;
     WIN_TRUST_SUBJECT    Subject;
}WIN_TRUST_ACTDATA_SUBJECT_ONLY,
        *LPWIN_TRUST_ACTDATA_SUBJECT_ONLY
```

请注意，这个结构包含了两个变量，dwSubjectType 定义了你将要检验的对象类型，你可以为大部分数据文件选择 WIN_TRUST_SUBJTYPE_RAW_FILE，或为可执行文件选择 WIN_TRUST_SUBJTYPE_IMAGE（其中包括 DLL 和 OCXs），Subject 结构指向你要检验的对象，其格式为：

```
typedef struct _WIN_TRUST_SUBJECT_FILE
{
      HANDLE    hFile;
      LPSTR    lpPath;
}WIN_TRUST_SUBJECT_FILE,*LPWIN_TRUST_SUBJECT_FILE;
```

正如你所看到的，这个结构中的两个变量分别指向文件和文件的路径。在大多数情况下，你将在浏览器或其它应用程序的缓冲文件夹内找到这个文件，例如，Internet Explorer 3.0 将它的缓冲内容保存在 Internet 临时文件中（数据文件）或 OCCACHE（可执行文件）文件夹中，它们位于 Windows 的主目录下，而 Netscape Navigator 的大部分 ActiveX 控件将临时数据保存在 ActiveX 控件缓冲文件夹（也在 Windows 的主目录下）。

那么 Windows 到底是如何检验文件呢？WinVerifyTrust()在执行中，首先寻找 PKCS#7 标志的数据结构，它是在为控件签名过程中创建的（我们在下一节中将要讲述）；然后再寻找一系列 X.509 证书，在当前实施的 Windows 软件发布信任供应商中，你必须伴随着软件发布的公开密钥，再提供一个私有密钥（我们将在下一节中讲到）。在未来，Windows 软件发布信任供应商也将寻找 X509.3 扩展定义的密钥使用限制和其它属性，如果检验到 PKCS#7 数据结构和 X.509 证书都是正确的，WinVerifyTrust 将给你的应用程序返回一个成功的信息。

Internet 部件下载机制

你也可以把本节命名为“对代码进行签名”，这个过程实际上包括四步，你可能要多次重复这些步骤才能得到正确的结果。第一步是创建一个在本地 Web 服务器上使用 ActiveX 控件的文档，你把这个服务器当作一个 Internet 站点来访问，我们在第 10 章中已经讨论了它的详细过程，这里就不再介绍了；第二步是删除控件的注册信息，许多编程语言为你创建的控件自动注册，即便不能自动注册，你也要作为测试的一部分来注册它，去掉控件注册后，所有的注册项被

删除，这些项是 WinVerifyTrust()用来寻找控件的一个本地拷贝时使用的，当它找不到注册项时，你所看到的屏幕将与用户看到的一样 第三步是对你的 ActiveX 控件进行签名 ;第四步也是最后一步，就是访问用来测试的 Web 站点并查看 Web 页，它将下载这个控件并允许你测试签名的过程，你可能需要对后三步进行多次测试才能最终确认签名过程的正确性。

去掉控件的注册信息

在 SYSTEM（或 SYSTEM32）目录下，有一个叫 RegSvr32 的小程序，它用来负责增加 ActiveX 控件在注册表中的登记项，使用命令行开关-U，它也可以取消控件相应的注册项，如果你想要取消第 10 章中创建的基本控件的注册项，那么你可以在命令行输入 REGSVR32 OCXEXMPL.OCX -U。成功删除注册项时，你会看到如下图所示的对话框。



签名你的控件

对你的控件进行签名也分为四个步骤，第一步是创建一个 X.509 证书（我们在前几节中从安全的角度讲述了这个证书），你将使用 MAKECERT 应用软件来创建一个基本的证书。这个应用软件的文档在 SIGNCODE.TXT 文件中，

你可以在 ActiveX SDK 的安装软件的 BIN 目录下找到这个文件。

注释 Visual C++5.0 提供了 MAKECERT 这个应用软件，你将需要它来签名你的控件，然而，在安装中并不加载这个应用软件，在 Visual C++ 安装光盘的 Cab&Sign 目录下，你将找到它的所有文件，确认你已经把这些文件拷贝到了硬盘中（它不能在光盘中运行）。以下处理过程假设你使用 ActiveX SDK 获得了最新的工具，但这里使用的是 Visual C++5.0 提供的工具。

```
MAKECERT-U:AKey-K:AKey.PVK -n:CN=MyCompany -d:A-Company TESTCERT.CER
```

这个命令行是用来创建一个随机的公开/私有的密钥对，并将这个密钥对和便于识别的名字联系起来。我们也创建了一个私有密钥文件（PVK），它保存私有密钥的备份，你创建好一个证书后，必须将它和一个基础 X.509 证书结合起来，这个证书来自诸如 VeriSign 的 CSP。由于现在还没有任何 CSP 来处理这类证书，微软提供了一个测试证书，你可以通过调用 ROOT.CER 来使用它，它和其它文件都在 BIN 目录下。

我们将使用 CERT2SPC（certificate to Software Publishing Certificate）应用程序来将两个证书加入到 PKCS#7 签名块中，这个对象实质上充当包括在签名对象中的证书的句柄，一般只有两种证书，以下我们使用其中的 CERT2SPC。

```
CERT2SPC ROOT.CER TESTCERT.CER TESTCERT.SPC
```

在这个例子中的三个参数分别是微软提供的基础证书、我们在第一步中创

建的证书以及用来存储 PKCS#7 签名块对象的新文件名。现在你已经有了一个完整的证书，在第三步中，你将把它加入到 Active 控件中，即使用以下命令行。

```
SIGNCODE -prog OCXEXMPL.OCX -spc TESTCERT.SPC -pvk MYKEY.PVK
```

这个命令行实际完成了很多任务，它做的第一件事是创建一个加密摘要的映像文件（在此例中是 OCXEXMPL.OCX），然后用私有的密钥标志它，这个加密的摘要允许客户对映像文件的当前状态和它接收到的文件状态进行比较，客户将使用这个比较结果来检测文件是否有任何形式的损害；下一件事，SIGNCODE 删除从 SPC 文件中发现的所有 X.509 证书，并使用 X.509 证书的序列号和已标志的加密摘要创建一个新的 PKCS#7 数字签名对象；最后，它将这个新的 PKCS#7 数字签名对象随同 X.509 证书一同嵌入到映像文件中。

考虑到 SIGNCODE 程序所做的大量工作，你将需要检验它的结果，这就是第四步要处理的工作。首先要检验数字签名是否正确地嵌入了文件，你可以使用 PESIGMGR 应用程序来处理，你所要提供的只是被检验文件的名字，如果获得一个成功的信息，便可使用 CHKTRUST 程序来检验可执行代码是否已用数字签名正确地表达了，这个程序与用户浏览器所做的工作相似，它首先检验数字签名对象，然后是 X.509 证书，如果两者都检查完毕，CHKTRUST 也将检验映像文件的代码，确认是否与数字加密摘要相一致。CHKTRUST 最后所做的处理是浏览器一般不做的，那就是检验多种证书之间的连接关系并确认独立的链最后终结在基础证书位置。

完成实际测试

检验已签名控件所要做的最后一步是实际使用该控件。你必须通过 Internet 连接来进行测试。简单地将控件移入你的 Internet 服务器中，确认它不再在你的本机上有注册，然后尽量使用浏览器来查看测试页。如果一切工作正常，当你下载这个控件时，你应该看到一个显示出来的证书，而不是一般的警告信息。

即便控件被正确地签名了，也还有一些东西会导致你的测试失败。可能发生的第一个错误是没有正确格式化<OBJECT>标志，这个标志在第 8 章中我们已经讨论过。请确认你的 CODEBASE 属性指向了控件所在的位置，另外，在你的机器上使用 URL 来代替目录位置，例如，如果控件位于 Web 服务器的 CONTROLS 子目录下，则 CODEBASE 属性使用如下的 URL：http://www.*mycompany*.com/CONTROLS/OCXEXMPL.OCX。

在本机做好签名后，你可能想双重检验你的控件。虽然控件在签名过程中受到损害的情况很少，但还是存在这种可能性的，如果你第一次在远程位置测试时，控件工作不正常，那么请确认控件在你的本地测试页上工作是否正常。

14.5 安全标准

在前几节中，我们集中介绍了为 Internet 编写代码将要面临的所有问题，这些问题确实存在，但你没有必要一个个地处理它们。在你阅读本书时，标准化组织正巧提出了保护数据的方法。你所要做的就是学习由这些组织提出的管理 Web 站点的方法。采用这些方法有两方面的好处：首先，你不必重新摸索和创

造这些标准；其次，你所采用的安全方法能够与其它站点采用的方法相一致，它减少了用户在学习中所走的弯路，可以使其他编程者使用那些已开发好的工具。

Web 链接 如果你想寻找有关 Internet 安全标准的最新信息，请访问 http://www.w3.org/pub/www/security/，这个 Web 页虽然只提供了概括性信息，但它能让你连接到其它站点上以获得更多的信息；另一个好的站点是 http://ww-ns.rutgers.edu/ww-security/reference.html，它放置了 Rutgers 大学网络服务中心提供的 WWW 安全会议资料，这个站点主要告诉你各种安全会议的联系方法以及摘要信息，而不是有关安全的详细规范；如果开发者想要获得商业角度的安全信息，可以访问 http://www.rsa.com/，RSA 站点覆盖了一个非常广阔的主题，其中包括 MasterCard 和 VISA 开发安全信用卡事务的当前状况；通过 ftp://ftp.isi.edu/internet-drafts/lid-abstracts.txt，你也能看到 IETF 产品的当前状况。

下面，让我们看一看 Internet 的各类安全标准，这些标准有的已经实现，有的正在成为标准。表 14.1 显示的是在编写本书时已有的标准或标准的草稿，在你阅读本书时，又会出现更多的标准，令人惊奇的是，在 Internet 上的厂商标准可能是除浏览器技术外增长最快的一个领域，（浏览器技术的发展速度如此之快，以至于测试者都难以跟得上它）。你还将注意到，这里列出的主要标准不是来自微软或其它公司，而是来自下述两个组织之一：IETF（Internet Engineering Task Force，Internet 工程任务组）或著名组织 W3C（World Wide Web

Consortium）。IETF 已有很长的历史，它是最早为 Internet 工作的组织。W3C 对每一类要用于 Internet 的新标准负责，例如，微软现在正努力征得 W3C 对其 <OBJIECT>标志和其它与 ActiveX 相关的 HTML 扩展的许可。

表 14.1　Internet 当前的安全标准

标准	描述
分布式身份认证安全服务（DASS）IETF RFC1507	DASS 是 IETF 正在改进中的一个标准，它为在 Internet 上提供身份认证服务定义了一个试验的方法。这里认证的目的是确认发送消息或提出请求的人是谁。DASS 正在努力解决当前保密字设计中存在的许多问题，例如，现在还不能检验出发送保密字的人是否在模仿他人。由于 DASS 是在一个分布式环境中提供认证服务，所以它有着特殊的挑战，因为用户不止是登录一个机器，他们能在网络上登录所有能够想到的机器

续表

DSI（数字签名优先权）	这个标准由 W3C 制定，它用于克服通道级安全的一些限制。例如，通道级安全不能对文档或应用程序的语意做处理；因为所有的处理过程都发生在 Internet 上，而不是在客户或服务器端，所以通道也不能有效利用 Internet 的带宽。DSI 为传输签名定义了一个数学的方法，它实质上是对指定的个体或公司做唯一的表述。DSI 也为标识安全属性提供了一种新的方法（PICS2），为维护提供了一种新的格式（PEP）。DSI 标准也被建立在 PKCS7 和 X509.V3 标准中
扩展的 Internet 标记 SHTTP（EIT SHTTP）	它是 SHTTP 的扩展，它在当前的 HTTP 列表中增加了与安全相关的标志。目前还没有标准化组织支持这种技术，但以后将会有的。（IETF 最近成立的 Web 事务安全组织正关注着这一技术，你可以通过 http://www-ns.rutgers.edu/www-security/wts-wg.html 来进一步了解详细信息）

续表

通用安全服务应用程序接口（GSS-API）IETF RFC1508	这是个已批准的 IETF 规范，它定义了以通用方式支持安全服务调用的方法。使用通用接口允许在更广泛的平台提供更多的源代码级可移植能力。IETF 并没有把这个标准看成是这类标准的结束，而是一个开始，将来会有更多的规范和标准。然而，学会这些已有的标准将帮助你在各种安全实施方法中找到通用的线索
通用安全服务应用程序接口（GSS-API）C-binding RFC1508	这是个已批准的 IETF 规范，它定义了 C 语言中使用支持服务调用的方法，这个标准是最先基于 RFC1508 实现的标准之一
Internet 协议安全协议（IPSec）	IETF 最近成立了一个 IP 安全协议工作组来寻找 IP 安全性方面的问题，比如，在 IP 协议级无法对数据加密的问题。这个工作组当前正在研究覆盖面相当广泛的规范，以便最终提供更加安全的 IP 功能。在 http://www.ietf.cnri.reston.va.us/html.charters/ipsec-charter.html 站点你可以找到有关这个组织的更多信息

续表

JEPI（Joint Electronic Payment Initiative，联合电子支付协议）	它是由 W3C 创建的标准，JEPI 提供了创建电子贸易的方法。使用电子现金或信用卡进行交易，从客户到服务器间的数据传输将使用加密、数字签名、身份认证（密钥交换）来保证安全的交换。这是一个新兴的标准，它的一些条目，比如传输层安全（也叫私有），当前正通过 IETF 继续向前发展
Kerberos 网络认证服务（V5） IETF RFC1510	这是个已经批准的 IETF 标准，它定义了第三方认证的协议。Kerberos 模型部分基于 Needham 和 Schroeder 的可信任第三方认证协议和基于 Denning 与 Sacco 提出的第三方认证协议的修正协议。与许多 Internet 认证协议一样，Kerberos 作为一个可信任的第三方认证服务来工作。它使用了依赖于把共享的公共密钥与私有密钥结合起来的常用加密方法。Kerberos 着重于带可选服务器认证的客户身份认证
Privacy Enhanced Mail part I（PEM1）消息加密和认证过程 IETF RFC1421	这是个已经批准的 IETF 标准，它用来确认你的私有邮件依然保持秘密。实质上，它显示了一个你在加密邮件中采用保护方式的过程，这些过程在解密过程中用户是看不见的。这个标准运用了密钥和其它形式的认证管理。此标准的一些条款基于 CCITT X.400——特别是采用了邮件句柄服务（MHS）和邮件传送系统（MTS）中的某些条款

续表

Privacy Enhanced Mail part II（PEM2）基于证书的密钥管理 IETF RFC1422	这是个用于管理安全密钥的已批准 IETF 标准。它提供了基于公共密钥认证技术的基础结构和管理结构。IETF RFC1422 是 CCITT X.509 规范的增强改进版本，它为使用 PEM 提供了密钥管理基础的过程和协议，这是 CCITT X.509 中所没有的
Privacy Enhanced Mail part III（PEM3）算法、模式和标识 IETF RFC1423	这是个已经批准的 IETF 标准，它定义了加密算法、使用模式和 PEM 特定用法的标识。这个标准覆盖了与加密信息相关的四个主要领域：信息加密算法、信息完整性检测算法、对称密钥管理算法、非对称密钥管理算法（包括对称加密和非对称签名算法）
Privacy Enhanced Mail part IV（PEM4）密钥证书和相关服务 IETF RFC1424	这是个已经批准的 IETF 标准，它定义了验证密钥的方法，它也提供了一个与加密相关的服务列表，Internet 站点可能需要把它提供给最终用户

续表

安全/多用途Internet邮件扩展(S/MIME)	这是由多家公司联合提出的一个标准，这些公司包括Microsoft、Banyan、VeriSign、ConnectSoft、QUALOCOMM、Frontier Technologies、Network Computing Devices、FTP Software、Wollongong、SecurWare和Lotus，它最初由RSA Data Security公司开发，目的是为了使不同产品的开发者能使用兼容的加密技术创建消息传输代理（MTAS）。它实质上意味着，如果有人用Lotus产品发送给你一个信息，你能使用Banyan产品来读取它。S/MIME以流行的Internet MIME标准（RFC1521）为基础
安全/广域网(S/WAN)	在很多人眼里，S/WAN现在只是一点朦胧的微光。S/WAN是RSA Data Security公司提出的一个标准，IETF也有一个委员正在研究这个标准，RSA准备将IETF的IPSec标准也纳入S/WAN中。S/WAN的主要目标是允许各公司最佳地混合和匹配防火墙与TCP/IP栈产品，以创建基于Internet的虚拟专用网（VPNs）。这两种产品当前的方案通常是将用户封锁在单一资源内

续表

SHTTP（安全的超文本传输协议）	这是 Open MarketPlace Server 公司当前使用的加密数据传输技术。在功能上它与安全套接层（SSL）非常相似，而最大的不同是这种方法只应用于 HTTP。现在还没有标准化组织支持这种标准，不过将来会有的。（IETF 最近成立的 Web 事务安全组织，正关注着这方面的技术）
SSL（安全套接层）	SSL 是一个 W3C 标准，它最初由 Netscape 公司提出，目的是为了让客户至服务器端能从协议层传输加密信息。套接口允许在高层协议如 HTTP、NNTP 和 FTP 上进行低层的事务加密。这个标准也制定了客户和服务器的身份认证方法（客户站点的身份认证是可选的）。通过 http://www.netscape.com/info/security-doc.html，你可以了解这个标准的详细信息
WWW 上的统一资源识别（URI）IETF RFC1630	URI 是 IETF 正在开发的标准，目前，资源名和地址是纯粹的文本。URL（统一资源地址）实际是包含了地址信息的 URI 的一种形式，这个地址在 Internet 上映射到一个指定的位置。URI 能提供将 Internet 对象的名字和地址编码的方法。实质上，要访问一个私有的站点，你应该知道它的编码名字，而不是纯粹的文本名

注释 MTS 是个有多种意思的缩略语，在本书中只用到了两种：微软事务服务器和邮件传输系统。除了第 14 章中代表邮件传输系统外，一般

都是指微软事务服务器。

Web 链接 表 14.1 中为你介绍了很多由 IETF 创建的与安全相关的标准。大部分 IETF RFC 文档能在 http://ds.internic.net/rfc/上找到。在http://www. ietf.cnri.reston.va.us/html.charters/上，你也能看到IETF当前工作组的列表，这些工作组协助创建 Internet 上使用的标准。

表 14.1 中的安全标准描绘了 Internet 最终的安全蓝图，将它牢记在头脑中是很重要的，所有这些标准真正覆盖了客户和服务器间的连接。你还能在客户、服务器端增加其它安全策略。这些标准中的大部分还没有覆盖 Internet 的附加产品，比如防火墙等。你的公司能够增加这些附加的安全特性，并最终使得黑客很难攻进你的系统。

Web 链接 如果你还没有足够多的安全信息的话，可以花点时间查看一下 IETF 的情报站点，目前的两个站点是 gopher://ds1.internic.net/00/fyi/fyi8.txt（站点安全手册）和 gopher://ds1.internic.net//00/fyi/fyi25.txt（网络信息检索状态报告）。

谁是 W3C

在我们做进一步学习前，让我们大概了解一下 W3C 组织。W3C 最早出现在 1994 年 12 月，当时，它批准了 SSL（安全套接层）；在 1995 年 2 月，W3C 也支持了 Internet 上的应用级安全。它当前的项目是数字签名协议，W3C 于 1996 年在巴黎发表了该协议。正如你所看到的，W3C 是一个专注于安全需求的标准化组织，它的其它一些职能也是从这个角度发展起来的。

技巧 如果立即使用安全套接层（SSL），将还有许多安全方面的顾虑，因为一些计算机黑客在特殊情况下有可能破解你的密码。美国.政府限制出口加密技术中的密钥长度不能超过 40 位，所以计算机通过尝试 2 的 40 次方的密钥组合，便可以破解这个密钥；美国本土使用的加密程序只能使用 128 位密钥，它意味着一个人要等待计算机做 2 的 128 次方的字母组合才能破译它（这恐怕一辈子也等不到）。那么你将如何判断交易的安全等级呢？Netscape 使这个问题变得非常容易，你只需查看一下屏幕左下角的钥匙：一把断开的钥匙表示没有加密；一把单排齿的钥匙表示 40 位的加密；一把双排齿钥匙表示 128 位的加密。Netscape 和 IE 都提供了文档属性对话框（在“文件”菜单中可以找到它），你可以简单调用这个对话框来查看指定站点提供了什么类型的加密保护方式。

那么我们为什么还需要其它的标准化组织呢？其主要想法是把业界的主要厂商集中起来，创建一个安全的 Internet 网络环境。另外，某些标准化组织——

比如 IETF——主要由自愿者组成，它们的动作过于迟缓，根本不能满足企业的当前需求。W3C 是在为增加新标准并尽快将其实施而产生的组织，W3C 最终提供了实际的标准和规范说明，并提供了采用这些标准的应用示范，以及使用这些标准创建应用程序所需的代码等。

W3C 所研究的另一个项目是 Joint Electronic Payment Initiative(JEPI)，一个将影响 Internet 商务交易的标准。它大胆地假设在英国、法国、德国和美国的客户都能访问同一个站点，并以完全相同的价格去购买同一个商品。但在此之前，它们必须解决一些问题，例如，你是否要询问客户提供一种付款的方式？你需要什么样的付款？你还必须考虑到纸币以外的东西，因为纸币还不能通过电子方式交换。信用卡提供了一种解决方案，但 W3C 也在探究其它方式，如电子支票（electonic check）、借贷卡（debit card）和电子现金（electronic cash）等。这只是其中的一个小主题，更大的主题是要为商品的价格创建一些与纸币无关的方法。例如，你如何告诉一个在德国的人，外套的价格是 80 美元？如果不给他们展示一美元是多少德国马克，他们是否也能知道？对美国人来说，他们若访问一个英国商店，则必须找到兑换英镑的价格。

那么 JEPI 将对安全性做些什么工作呢？它将交易作为一个整体来考虑。你的公司也许不用与一个想买外套的客户打交道，但有可能要与其它一些和贸易有关的问题打交道。大公司也许想提供一个更快、更有效的方法来将资金从世界的一个地方转移到另一个地方。通过 Internet 做电子贸易将最终解决这个问题，这当然不是眼前所能做得到的。如果你的移动雇员缺乏资金时，你该怎么办呢？你可能想到立刻发行一个信用卡，虽然你可以通过 Internet 获得一个日消

费的报告，但现在这样做是不明智的，因为 Internet 还存在太多的安全漏洞。而这些交易标准将帮助公司建立一些新的方法来传送敏感数据。

处理货币的标准方法

如果你的公司在 Internet 上出售产品（不论是有形的还是无形的），那你必须找到一个能在 Web 服务器上收钱的方法，就像在其它地方一样，你能在 Internet 轻松地出售装饰品，甚至还可以卖得更好，但你还得提供一些服务，例如，你要为客户提供一种电子支付帐单的方式，而不是通过邮件送传支票。有谣言说，可能至少有一家电话公司正在作 Internet 监听。小公司可能会发现 Internet 交易太麻烦，但一些中等和大型企业将在某些方面倾心支持它，银行和信用卡公司最终也将加入其中。事实上，通过在线填写的形式，你便能获得一个 Web Conductor VISA 卡（查看站点 http://www.conductor.com 以获得详细的信息）。这个特制的卡允许你在线查看状态和请求服务，在不久的未来，它将允许在线付款，以大大减少填写支票所花的时间。

当前有三种不同的公开方案来实现在 Web 服务器上处理现金，它们是 First Virtual Account、DigiCash 和 Cybercash。每种方案都各有千秋，也各有缺陷，你应该认识到的是，它们中没有一个是至善至美的，如果现在实施这些方案，你可能需要在某些地方做一些妥协。虽然有许多金融公司正在研究一些方法来确保金融贸易的真正安全，但恐怕目前尚无有效的方法。

从实施的角度看，First Virtual Account 方案更容易使用，用户通过电话报名加入一个 First Virtual Account 并接收到一个指定的帐户名，这个帐户名与信

用卡提供的帐户名毫无关系，支持这种方案的商家使用安全证书来接受这个指定的号码。每个交易都使用这个号码，而不使用真正的信用卡号码，这样防止了有人偷去你的信用卡并趁你不知道时使用它。商家通过 First Virtual Account 做交易，它能检验用户的事务，当用户同意交易，商家便接收到付款并开始运送用户所需的产品。你能在 http://www.fv.com 站点上看到有关 First Virtual Account 的更多信息。

DigiCash 取代了信用卡而使用真正的资金做交易，这意味着商家不必支付信用卡的追缴款。另外，DigiCash 还可以像 ATM 借贷卡一样工作，商家获得了真正的资金，而不用等待信用卡公司来按期支付，这需要用户在一个指定的银行有抵押，银行设置一个帐户，除了在线工作外，这个帐户就像是一个支票户头。当用户购买物品时，他或她使用“ E-cash ”，商家简单地进入 E-cash，便可获得真正的资金。显然，用户必须不时地添满自己帐户内的资金。你可以访问 http://www.digicash.nl/站点来获取更多信息。

Cybercash 是前面两种方案的组合，它提供了借贷和信用两种能力，有点像银行内有透支保护的户头。与其它两种方案不同的是，Cybercash 使用真正的帐户信息，对用户来说，这有点冒险，因为他们的帐户信息能在任何地方以非加密的形式终止。当用户购买东西时，Cybercash 弹出一个窗体，要求输入帐户信息，在信用方式下，用户要提供自己的名字和信用卡号码；在借贷方式下，用户要提供本地银行的名字和其它银行的帐户名。你可以在 http://www.cybercash.com 站点看到与此相关的更多信息。

Web 链接 不止是银行和信用卡公司在讨论在线交易问题，作为一家 Web

服务器公司，Open Market 也在制定基于 SHTTP 的在线交易方案，如果你使用了这个方案，那么它将充当银行和信用卡公司。你可以在 http://www.openmarket.com 站点找到与此相关的更多信息。

第 15 章　建立帮助文件

概述一般原则总是要冒风险的，特别是对一本书。程序员最头疼的任务是书写文档，但是比起写帮助文件来，那可算得上是美差了。当用户使用软件时，帮助文件做得很糟糕，就自然会去读随机文档，尽管后者也不见得好多少。大多数情况下，用户总是抱怨帮助文件只是做做样子，而不是真正对使用软件起到帮助作用，其实程序员已经耗尽心机了。

最近，帮助文件的做法引入了一个新的热点——基于 HTML，结果是更添乱了。Windows 的帮助文件机制尽管不完美，但它提供了一个完善的搜索引擎，用户可以自行查找想要的条目。其不足之处在于某些情况下软件更新太快（比如操作系统），而相应的帮助文件难以及时更改。基于 HTML 的帮助文件的好处在于易于更新。在向电子商务迈进的时代，灵活易变的帮助文件越来越重要了。

回想一下，我们会发现帮助文件与应用程序本身一样重要。如果用户面对庞大的程序不知从何下手，程序员即使挖空心思设计了一个得意之作，也不会得到如潮的好评。这时一切都成了无稽之谈：新增了好多功能，有什么用呢？用户根本找不到！这些足以说明优秀的帮助文件对一个软件来说是何等重要啊！

您该做的最后一步就是书写帮助文件了——想躲也躲不掉的任务！应用程序越大，帮助文件也就越长越复杂。通常 4 个小时的程序开发工作量需要 1 个

小时的建立帮助文件时间。您可以缩短帮助文件的编写时间，但切记不可偷懒。帮助文件中做的成功的部分是那些作为程序用户手册的那部分。我认识的某些人就使用同一个文件既做联机帮助、又做用户手册。

注 基于 HTML 的帮助文件易于更新，使用灵活，有些厂家用它来代替印刷的说明书。

注 帮助文件是应用软件的指南：帮助用户快速掌握软件的使用方法。

Web 链接 如何把帮助文件组合起来呢？本章主要阐述这个问题，这时可能要和其它程序员协作。有些站点专门讨论帮助文件，例如：comp.os.ms-windows.programmer.win32 和 microsoft.public.win32.programmer.tools 新闻组。还有一些站点也有讨论，如 microsoft.public.access.chat 新闻组专门讨论数据库应用程序的帮助文件。如果开发 Internet 应用，那么 microsoft.public.internet.news 新闻组很有帮助（在 Internet 应用中，基于 HTML 的帮助和 Windows 帮助都可采用，但前者更普遍）。甚至可以在一般的编程讨论组如 comp.programming 中加以讨论。

注 编写帮助文件首要的一步是列出大纲，就像编程一样。

以下各节将会帮助您解决许多编写帮助文件中遇到的难题。目前有许多编写帮助文件的技术，所以必须首先确定哪种技术最能满足您的需要。本章的第一节将会谈到两种技术，并给出各自选用的根据。显然，无法定义一种放之四海皆准的标准，但至少有些基本的出发点需要考虑。

像所有编程工作一样，当我们着手建立一个帮助文件时，最先考虑的就是

如何组织的问题。如果您的思路清晰明确，写起帮助来就会得心应手。事实上，您要做的就是把程序流程通俗化、条理化。

接下来就该考虑如何实施了。本章介绍了两种 Windows 帮助文件的实现方法。第一种使用 Help Compiler 加 RTF 文件实现，当帮助文件中有许多唯一特性，而且要快速查找时，这种方法很适用。第二种方法基于 Microsoft 的 Help Workshop，这种方法的好处在于省时易学。

只有将 Windows 帮助文件集成到应用程序中后，帮助文件才能发挥作用，本章的下一节将讨论这个问题。我们考查几种不同的方法为第 2 章中建立的应用程序编写帮助文件，您会看到如果要具有上下文相关帮助功能，帮助文件将变得非常庞大，如同小驹配华鞍，喧宾夺主！

Web 链接 也有相当一部分人放弃采用 Windows 帮助文件模式，转而使用 HTML 文件，就连微软目前也在它的部分产品中使用了这种方法。HTML 文件的主要缺点是搜索功能不强，格式不够规范；其好处则在于内容更新十分方便。如果想进一步了解这方面的情况，可查阅 HyperNews Forums Web Mastery Resource List，其网址为：http://union.ncsa.uiuc.edu/HyperNews/get/www/html/guides.html 这个站点中包含了许多内容，指导您将 Windows 格式的帮助文件转化成 HTML 格式。还可以参看 WWWWAIS.C 站点，网址为 http://www.eit.com/software/wwwwais/wwwwais.html，WAIS 是个让你在 Web 站点中加入搜索功能的程序。

基于 HTML 的帮助文件也需要最后的实施阶段，然而其实现方法更像是设

计一个 Web 主页的过程。在阅读本章之前，请先读第 8 章，从而对 HTML 概念有一个基本的了解，本章只讨论建立基于 HTML 的帮助文件时，有哪些需要特别考虑的地方。

本章的最后一节将演示如何把基于 HTML 的帮助文件集成到第 2 章所建立的应用程序中。尽管不是很难，但还是比集成 Windows 的帮助文件要复杂一些。另外，你会发现 HTML 在生成真正的上下文相关帮助方面存在一些限制，但可以采取一些迂回措施达到这一目的。最后本节还将讨论为什么有许多的厂家喜欢使用 HTML 做帮助文件——就是为了便于更新！

15.1 确定要创建帮助文件的类型

微软试图使人们相信基于 HTML 的帮助文件是有史以来最好的帮助形式，也是您开发新的应用程序时应该唯一考虑的选择。然而帮助文件形式的选择是由一系列因素决定的，而不只是为了赶时髦。

注释 寻求基于 HTML 的帮助文件已经成为商业软件的事实标准，原因在于它易于建立、减少了编辑使用手册的开销。然而少花钱并不等于这种文件就好使了。

事实上，许多微软的用户正在抱怨基于 HTML 的帮助文件对于微软的某些产品来说很不实用。当然也有一些情况则很适合用这种形式的帮助文件。那么，我们究竟怎样取舍呢？我们最好还是先对二者加以比较，以下就是选择 HTML

的原因：

升级能力　建立 Windows 帮助文件并嵌入应用程序后，帮助内容就相对固定了，因为一旦装入了用户的机器就不好再更改了。基于 HTML 的帮助文件驻留在某一 Web 站点上（包括 Internet 和 Intranet），所以一经更新，用户总是看到最新的内容。

减低了学习难度　随着 Internet 和 Intranet 的日益普及，企业中只需几个人掌握建立和更新 Web 主页的技术，专门负责维护就可以了，而 Windows 帮助文件则需要每个程序员都理解其编制规则，显然浪费了劳动力。

减少了与语言＼特殊需求相关的问题　有许多种方法使得 HTML 帮助文件支持多种语言、面向特殊应用，而不必考虑 Windows 帮助文件的影响。可以先询问用户想用哪种语言，按照回答提示用户转入相应的站点去查阅。对文种支持的自动化使得不必在每个应用程序中再考虑文种的重定向问题。应用类似的技术也简化了其它一些需求，比如用户需要支持大字体等。

增强了定制化的机会　HTML 的开放机制使得定制帮助文件非常容易。Windows 帮助文件中，所有的图形都存在于编译好的文件之中，而基于 HTML 的帮助文件则不然，可以随时提取其中的图形，这就意味着可以按照用户的要求进行度身定制，如更改公司徽标啦、反映企业形象啦等等，一切信手拈来。

然而这并不是说 HTML 帮助就无所不能了！事实上，它也带来了一些新的问题。以下就是使用 HTML 帮助文件时所不具备的特性：

安全性　使用 Windows 帮助文件时，可以隐藏一些信息只让管理员可以查阅。而 HTML 帮助文件都是文本形式，所有人都可以知道是否有管理员用户信息存在，并试图访问这些信息。

大小　Windows 帮助文件可以进行压缩，通过减少磁盘碎片来节省硬盘空间；而 HTML 帮助文件则无法压缩，磁盘空间将是个问题。

远程访问　Windows 帮助文件就在本地硬盘上，随时可用。而 HTML 帮助文件的可取之处也正是它的致命之处：如果用户无法访问 LAN 或 Internet 时怎么办呢？这时本机上拷贝的帮助文件即使已经过时，也变得异常宝贵了。

没有注释＼降低了书签支持　用户具有向帮助文件中增加注释的功能非常重要，然而这也是 HTML 帮助文件所不具备的。理论上可以允许用户修改原始的 HTML 文件，但比起 Windows 帮助文件来要复杂多了。相似地，即使只是在 HTML 帮助文件中加入一个“收藏夹”特性（类似于 IE 中的收藏夹），用户也只能保留该主页的地址。而 Windows 帮助文件可以随心所欲地在帮助文件中做标志，充分发挥书签的作用。

减低了检索能力　前面我们已经谈到了这个问题。Windows 帮助文件可以很方便地查找指定的单词。关键字的应用使得用户可以建立自己的查找标准。而 HTML 帮助文件不说是绝对不行，也是很难做到这种查找的，肯定不能查找单个字词或关键字。

说到这儿，您可能已经在权衡两种帮助文件的利弊了。的确它们互不相同，各有所长。我们选择的依据何在呢？您可能还出于一些我没有讲到的考虑来作

出自己的决定。

15.2 组织帮助文件

设计帮助文件有许多方法，如何组织是首要任务。组织的好坏是使用难易的决定因素。例如：面向任务的帮助文件可以指导用户迅速地完成任务，当然其前提是程序设计者很清楚自己的应用是干什么的。而面向菜单的帮助文件可以指导用户快速查找菜单条目。

帮助文件的类型也会影响其组织形式。例如：Windows 帮助文件提供了强大的查找能力，这时您就会更多地用到面向任务的组织形式，以便用户能够随时查找命令信息。此外，使用 HTML 很难建立教程性质的帮助文件，采用菜单流控制的组织形式则相对比较容易，因为新增菜单项比在 Windows 帮助文件中实现还容易。

建立大纲

着手建立帮助文件的最重要一步就是编写大纲。大纲应反映出用户的需求，这有助于从一开始就组织好结构，从而把写作重点放在具体内容上。下面是我个人的一些经验，供大家参考：

菜单流　菜单流技术以应用程序的菜单系统为中心，从它开始并作为主线。先将所有菜单项列成层次清晰的结构化条目，再列出所有对话框；每个对话框下要包含其中用到的所有控件；最后，列出主窗口及其组件。

使用这种组织方法有利于用户快速定位到应用程序的某一具体条目，缺点是用户往往只看到该条目的含义，却还是无法完成想要实现的任务。因此，这种方法适用于用户已经有一些相关经验的情况——只需要告诉他们本应用程序的用途，不用再从基础一一教起了。这种帮助文件很适合于实用程序，因为这一类型程序的用户购买该软件时目的明确，他们知道买来做什么。也适用于配置程序模块，这时的问题直接了当，但用户还是需要一些帮助的。

任务　我接触到的大多数用户并不关心当今程序中新出现的“gee whiz”特性。用户只是希望在最短时间内能最快地完成工作，特别对那些整日在公司里忙碌于琐事的职员，他们没有时间去学习如何做某件事，你只能告诉他们。这时，这种帮助文件就很有用了。先列出用户可以实现的所有任务，再一一加以解释。建议您从任务本身的解释开始，用户可以完成什么功能呢？再告诉用户采取什么步骤完成它。这些用户如果对计算机一无所知，而且毫无兴趣倒反而好了！这种技术适用于数据录入和任务固定的应用程序的情况，而对于字处理和其它综合性的应用程序就不能胜任了。

通用菜单／任务　有时所开发的应用程序执行许多任务。例如：一个字处理程序不只是一个目的，而有多种用途。如果此时您宏篇巨箸地逐一解释每项功能怎么使用，对您和用户都是一件残酷的事。我常用的做法是：对各种任务先进行分类，再对每类给出常规指导，而不是对每项任务加以解释。对菜单系统给出总缆介绍也值得推介，这样用户就能很快

了解应用程序的用途而不必陷于揣摩编者的构思之中了。由此引发了“气球帮助”的大量应用，这种气球是一种标志，鼠标在某一控件上停留几秒钟时它就会出现在鼠标指针旁边。

参考　编译程序和自带宏语言的应用程序经常使用参考手册类的帮助文件。这类文件是从控制的角度而不是具体结构上分析程序的。对一个字处理程序来说，按字母顺序列出宏命令清单就足够了，因为无法预测用户会怎么使用这些宏。通常还会具体描述这个命令，详述每个参数的含义，并给出一些用法的例子。显然也可以加入其它信息，如：提示不同版本的特殊性。用户常感到有用的信息有：“如果想做X，请使用此命令；否则请使用命令Y！”，这非常有效。

教程　这是一类特殊的帮助文件。只用于教给初学者，实质上是要告诉他们如何使用您的程序。多数程序员觉得：当用户毫无经验时，这种帮助文件非常有用。用户只需稍懂计算机使用常识就可以了。我发现适用于使用这种帮助文件的情况之一是数据录入程序。一般我先对某项具体任务给出一个简要的解释，再给出一些使用问答。如果能对实际应用进行抽象，给出交互式的帮助会话也非常可取。可以用帮助文件的宏命令来实现，它指导用户输入正确的操作。这一类型帮助的不足之处在于：大量的主题难于组织在一起，而且用户经过长时间的操作可能还搞不清楚到底要做什么。费时费力地做这种集成还不如努力说服公司雇一、二个培训人员呢！

功能区　一些应用程序提供这种帮助是因为它们特殊的用法。比如CAD

及其它的绘图程序，专门开辟有这样的功能区。CAD 程序中有绘图命令组、颜色模板控制命令组及图形大小形状控制命令组。将许多命令分组，有助于用户快速查找。原因很简单：用户绘图时只关心绘图命令，等到要进行润色加工时就只关心颜色啊、效果啊等相关命令。

Web 链接 如果您还是觉得编写帮助文件太麻烦了，可以使用现成的生成帮助软件。SheepNet 的站点 http://www.sheepnet.demon.co.uk/helpfile.htm 就提供了建立高质量帮助文件的方法。当然您还是需要花时间学习怎么使用，而且要想最终给出一个很好的帮助文件还是需要下一番功夫的。

建立脚本

注 帮助文件一般保证在 800 x 600 的显示方式下不超范围。

大纲建立好以后，就要补充血肉了！这时，也有一些规则应该遵循：首先，尽量使一节帮助的内容只占一屏（这种“一屏规则”对 HTML 帮助文件尤其重要，原因在于它不支持书签和注释，用户在第一次找到信息后翻到其它页再想快速查找先前看过的页面就不那么容易了）。用户可不愿意为了查找一个信息来回翻页。大多数情况下，可以将大块的帮助文件分成多个子标题，以便于查阅。在目前的硬件条件下，一屏的大小为 800 x 600（不久可能是 1024 x 768）。虽然有了更大尺寸的显示器，也还会有很多人用小的显示器。

技巧 大量新型笔记本电脑使用 1024 x 768 的显示方式，而老一些的用 800

x 600，甚至 640 x 480。在编写帮助文件时，要牢记这些条件。

“一屏规则”也有例外，例如：不想把一个过程分成几个子标题。谁碰到这样的帮助都会很恼火：像“建立邮件合并主文档——参见标题 3A 来实现”，您说烦不烦？类似的指南中的问题也常见于一些日用品的说明书中，比如音响的说明书。人们可能都听过这句话：“这东西读起来怎么跟音响说明似的，晦涩难懂”。

当不便于分开的时候可能也会打破“一屏规则”。例如：为了解释的直观性可能要加进图形，这时一屏就放不下了。多数情况下复杂的过程要调用多个屏幕，不要因为过程的复杂性而不使用子标题，只要有可能就尽量简化。所要做的是在过程的完整性和分割性之间取得平衡。要保证没有子标题的屏幕也同样容易阅读。我们常常把一些信息放在子标题下，这样用户看到的是一屏完整的信息，想阅读具体内容时只需再点击个别标题就行了。

以上我所说的可以总结为一句话：建立脚本。可能要做的工作像是写本书，但又不能用书的格式。不需要像书那样一个主题紧接着另一个（如果真把帮助文件做成书的形式不知要激怒多少人！因为有些主题只是介绍并没有实际意义），而要求每个主题都实有所指。当然，主题之间加入连贯性将有助于用户使用。

打脚本只进行了全部工作的一半，接下去就要将脚本变成 Windows 帮助引擎或 HTML 可识别的格式，从而用户才能使用。这时要将大纲分屏，并在需要的地方加进超文本链接。也有一些帮助文件编译器需要类似 make 文件的东西，用来告诉编译器在帮助文件中都包含哪些文件，并且可以用按钮进行选择。

以下各小节将完整介绍从脚本生成帮助文件的过程。之后，再介绍如何为用 Visual C++开发的应用程序制作具有上下文相关功能的帮助文件。那是建立帮助文件的最后一步。在把帮助文件放入应用程序之前应该确保所有的准备工作都已经到位。

高级技巧

词汇表的重要性

我们这些整天坐在机器前面编程的人随时会学到许多新的术语，这些术语就像是每天的调味品，而不会成为交流的障碍。常常自己不知不觉中就使用了没有专门去学习的术语。

而对多数用户来说计算机只是个半熟脸，不过是工作中用到的工具而已。（我的一个好朋友甚至把她的计算机叫‘死对头’，多好的形容词！）对这些人来说，新的术语就是障碍了。充满了晦涩难懂术语的帮助文件比没有帮助还糟糕，因为用户费劲读了半天啥也没弄明白。

我们必须面对这一事实，写帮助时不可能不用到术语，我们这个行业几乎每天都在产生新的词汇，只能尽量避免使用太专业的行话，却不可避免地要用到标准的计算机词汇。

一般在提交正式稿以前要打草稿，换句话说，先写个提纲，再逐步完

善。我想要说的是一个简单的书本的格式。这时，可以请一、二个从没有看过的人读一遍，记下他们不明白的词汇，这些人最好是非专业的，这样得到的信息才更真实。做完以后，把这些词汇汇编成表，就是一份很好的词汇表了，比你自己总结的要实用得多。

最后一步，将生词的每次出现都和词汇表进行热链接。这样，用户不知道生词的含义时只需单击链接处就可以了，Windows Help 或 HTML 浏览器就会自动转入词汇表的相应入口处。

15.3 使用 Microsoft Help Compiler

本章我不想介绍 Microsoft Help Compiler 的全部使用过程，因为建立帮助文件只需用到部分命令就可以了。而且，这个软件本身的帮助文件做得非常优秀，是程序设计人员学习的典范。我们要给出一个简单的例子，教您如何用标准字处理软件和 Help Compiler 组织帮助文件。目前为止，市面上还没有更好的工具，我们只能这么做。这也是建立帮助文件最灵活的一种方法，因为设计者对内容有最大的控制权。

注释 本节只讨论 Windows 帮助文件，如果想了解 HTML 帮助文件的技术，请参见本章以后的介绍。

Visual C++在建立帮助文件前还必须多做一步，所幸的是，这一步会为以后减少工作量，特别是在开始做 RTF 或 make file 之前。用 VC++的 BIN 目录下的

MakeHM（制作帮助映象）程序建立一个帮助映象文件。（对于 Visual Studio 的用户，该程序放在 Program Files\Microsoft Visual Studio\Common\Tools 目录下），命令格式如下：

```
MAKEHM    ID_,  HID_,  0X10000  RESOURCE.H  MY.HM
```

这样就会得到如程序列表 15.1 所列出的应用程序的帮助标志符，用于完成在 VC++和帮助文件之间的连接。现在，我们只需知道在帮助文件中将包含这些标志符，而且在帮助文件的脚本中也会用到。

程序列表 15.1

```
HID_CANCEL_EDIT_CNTR          0X18000
HID_CANCEL_EDIT_SRVR          0X18001
HID_UNDERLINE                 0X18006
HID_STRIKETHROUGH             0X18007
HID_BOLD                      0X18008
HID_ITALIC                    0X18009
HID_FONT_DIALOG               0X1800A
HID_VIEW_FORMATTOOLBAR        0X1800B
HID_FORMAT_FONT2              0X18010
HID_HELP_CONTENTS             0X18012
HID_HELP_WHATSTHIS            0X18013
```

注释 对编辑软件没有要求，就连设计程序环境中的编辑器也可以使用；唯一的要求是脚本文件必须是 RTF 格式的。所幸的是，RTF 实际上就是带有一些特殊格式命令的 ASCII 码文件。使用编辑器时要注意是否支持 CRichEdit MFC 类标准，因为有些老版本不支持某些 RTF 命令。尽管 Visual C++ 5.0 及其以后版本提供的 CRichEdit 还不像标准的编辑器那样专业，但对建帮助文件来说还是可以很好地工作的。

Microsoft Help Compiler（Microsoft 帮助编译器）至少需要两个文件：帮助脚本文件和 make 文件。我通常使用 Microsoft Word 来编写脚本，因为它支持所有的 RTF，而且我用得很熟。首先要将帮助文件分节，每节之间插入一个分页符，在 RTF 文件中，使用\page 语句进行分页（我在讲解过程中会介绍一些 RTF 语句，以供大家参考）。

Web 链接 如果微软的软件还不能满足您的需要，可查阅以下站点：http://win95.daci.net /webwhelp.htm，该站点提供了大量的 Windows95 的 WinHelp 工具和实用程序，例如：Help Maker Plus 程序，专为替代 Microsoft Help Compiler 而设计。另一个很好的站点是 Winhelp WWW Index，地址为：http://www.hyperact.com/winhelp/FTP_and_file_Archives/index.html，虽然该站点不包含直接的信息，却给出了大量的有用链接地址，如：指向 RTF 规范说明的站点，指向 SHED 规范说明的站点等等，还有指向其它帮助文件编译器和设计程序的站点。

接下来就是为每个标题加入一个或多个脚注了，许多超文本链接函数要用到脚注。例如：索引中的查找单词就是由它而来；还用于在词汇表和帮助文件之间建立链接。表 15.1 给出了一些脚注，这些都是常用的。还应该查阅一下 Microsoft Help Compiler 的文档，常会有新的补充。

那么脚注中有哪些内容呢？这要依脚注类型而定。例如：使用＃脚注，表示要加进超链接。使用前要仔细阅读使用规则，一定要注意脚注名的唯一性。名字最好是描述性的，否则自己也记不住。与变量一样，超链接的名字中不能含有空格。在 RTF 文件中脚注表示为：<footnote type>{\footnote<text>}。

然后要对这些含有标志符（＃）的脚注进行编译。有了这张清单，就可以顺利地组织帮助文件建立超链接了。我们先考查一下标准的帮助文件：绿色的文字表示超链接，实际它们是在双下划线或可点击的标识下的文字（RTF 中分别为/uldb 和/strike）。所以建立超链接的第一步就是进行这项工作。双下划线后是超链接的标题标识符，用隐含文本书写（RTF 中用/v 语句实现），这与＃脚注中的标识符相同。

这时还有一些事情要决定，比如：是否在帮助文件中加入图形。在适当的位置加入图形会大受欢迎，还可以加进声音和多媒体效果，但我觉得最好是在对这些技术应用娴熟之后再这样做，起初还是不用为好。还必须决定在 make 文件中放入哪些内容。以下的章节详细叙述了完成帮助文件的实施步骤，我还会解释各种可用的选项，以及对不同类型的帮助文件我是怎么实现的。

表 15.1 Microsoft Help Compiler 中的标准脚注样式

脚注类型	用途
*	最后机器上可能会留有许多 RTF 文件，但其中所包含的主题不是每个帮助文件都能用到的。举例来说，我为一个用于在线问候的通讯程序做了一个帮助文件，多数用户感觉不错，当他们在线看到陌生的姓名缩写时，借助帮助能很快加以识别。这是用含有多个主题的一个 RTF 文件实现的。这时又要为其它实用程序编写帮助文件，前面文件中除个别主题外，其它主题都适用，该怎么办呢？就用这个脚注标识符，它定义了一个建立标识，与后面将介绍的帮助工程文件（HPJ）配合使用。Microsoft Help Compiler 先检查帮助中要包含的主题，再检查 RTF 文件中的建立标识符。这个标识符必须做为主题的第一个脚注。建立标识是不区分大小写的，但我总是用大写，这样即使 Microsoft Help Compiler 以后改动了也不会受到影响。典型格式如下：*{/footnote BUILD_TAG}
#	用于标识主题的脚注。可以将它看成 GOTO 语句中用到的标号。当用随后介绍的技术跳转到这种主题标识符时，Windows Help 的控制也就转到这儿了。它属于帮助文件中超链接的一部分。超链接可满足多种需要，包括菜单和与词汇表建立连接。它也是不区分大小写的，我还是用大写，原因同上。典型格式为：#{\footnote SOME_LINK}
$	用于建立主题标题的脚注。帮助主题的标题出现在窗口的正文上方灰色区域内。也会在“主题查找”和“历史记录”对话框中出现。可以是所有文本。比如：${\footnote 这是一个标题}

续表

+	有时需要在一系列帮助主题间实现直接转移以方便用户使用，例如：一个长过程即使勉强放在一个窗口也不便于使用，改进的办法是将这个过程用几个可相互转移的窗口来表示。‘ + ’脚注就是用于这个目的的，可以把它称为浏览序列标识符。它激活了帮助窗口中的两个浏览按钮：<<和>>。Windows 允许使用任何标识符做为浏览顺序，它自动地对标识符以字母序排序；总是用页序号排序。例如：+{/footnote Page:1} 表示该序列中的第一页。要注意对每一个标题只能用一个序号 必须在 HPJ 文件中加入 BrowseButton 宏命令来使浏览按钮起作用。Windows Help 在帮助文件初始化过程中查找这个宏命令。本章中‘ 建立 make 文件 ’ 一节将讲述加入的方法 这个标识符是用来方便使用的，用户用它可以将长文分段而不会引起岐义。我发现在显示一个多页图形时也很有用，如层次结构图。例如：某帮助文件包含有 Novell 论坛的结构图，长度超过１页，用这个标识符用户就很方便地跨页阅读了。它也适用于参考手册型的帮助文件——它能够从一个命令转到另一个命令。它的应用范围几乎没有限制

续表

K	帮助文件的查找能力依赖于关键字脚注。在帮助文件中为每个主题及其子题目定义一个或多个描述字。我总是犯选过多关键字的毛病。如果用 Windows 95/98，关键字会出现在‘查找’窗口的‘索引’页。关键字可以含有各种字符，其中可以包括空格。Windows 还可以识别关键字的大小写，这样您就可以尽量描述清楚主题的含义，用户也就容易理解了。每个主题中可以有多个关键字，关键字之间用‘；’隔开。 在 DOS 的帮助编译器 HC31 中有一个缺陷，如果使用这个脚注会带来一些麻烦。在 RTF 文件中，这个脚注和其后的文字之间必须插入一个空格，否则它就显示不出来了。而且，如果关键字是以 K 打头的，更得多插一个空格或分号。 使用该脚注的一个实例为：K{/footnote Control; Exit Pushbutton; Leaving the Program}。这时用户能够找到同样的帮助主题：Control, Exit Pushbutton, Leaving the Program。在建立 RTF 文件时维护关键字的有序性有助于使复杂的帮助文件保持一致。一定要确保主题每次出现用的都是同一个关键字。例如：如果某处用到了“Control”，另一处也必须用同样的形式，千万别用复数或其它形式。如果帮助文件前后一致，用户很快就会用顺手，您也就会少很多麻烦，否则就等着到处答疑吧！

续表

@	如果一个程序没有注释会怎么样？当要增加新功能时您将无法指出哪些是原来就具有的。您看看本书中的例子，我总是给出许多注释，因为多年的编程实践使我充分认识到了注释的重要作用。随着帮助文件的复杂化，您很可能会忘记为什么加进了宏命令或其它的做法。标识注释的脚注解决了这个问题。它的用法跟向帮助文件加注释一样，区别在于大部分情况下这些注释不可见，除非以脚注视图打开它（当然前提是：文件是用标准编辑器建立的）。该类型脚注的典型格式为：@{/footnote This is a comment.}。不用说，帮助编译器根本不理会这种脚注，所以您什么都可以往里放

给帮助文件添加特殊效果

可以对帮助文件做许多润色工作。我最常添加的效果就是图形。例如：可从实际程序中抓取屏幕样图，用 Hotspot 编辑器(SHED.EXE)在图上定义热点，该编辑器支持 BMP、DIB、WMF、SHG 等多种图形文件格式。一般含热点的文件都使用 SHG 格式。

技巧 对教程类的帮助文件我最喜欢加入的图形是“回答”按钮。它们看上去就像标准的 Windows 按钮一样。用户看到提问作出回答后，按下这个按钮，就会得知操作得对不对。这个方法也可以应用于其它控件，Windows 标准帮助中没有提供这么强的功能，它提供了：{BUTTON [LABEL],Macro1 [:Macro2:...:MacroN]}的命令用以建立标准按钮。

Label 内容是按钮上显示的文字，宏命令则是实际执行的动作。

下面我们花点时间讲解一下 Hotspot Editor 编辑器。图 15.1 是它的典型界面，这里我打开了一个后面也要用到的例子程序窗口，该程序用于测试帮助文件。目前我们只关心主窗口的格式。我打算为帮助文件的每个控件设置热点（hotspot）以方便用户查找，不能用 Hotspot Editor 建立一个新图，真想那样做只能用其它图形软件；Hotspot 只是按照用户设想的动作在相应位置作出标识。当帮助文件中的光标由指针形状变为手指形状时，你就会看到 Hotspot Editor 的效果。

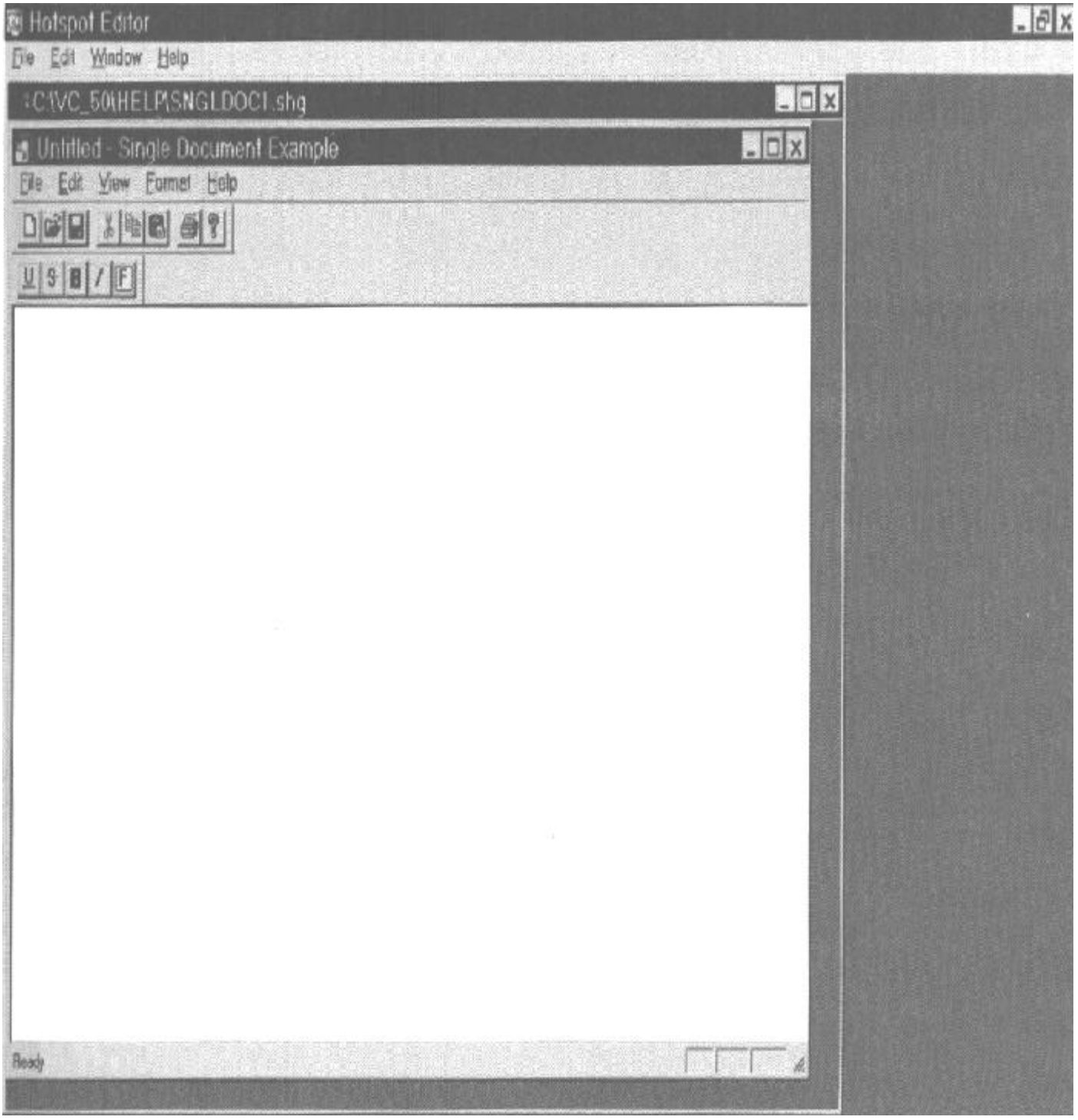
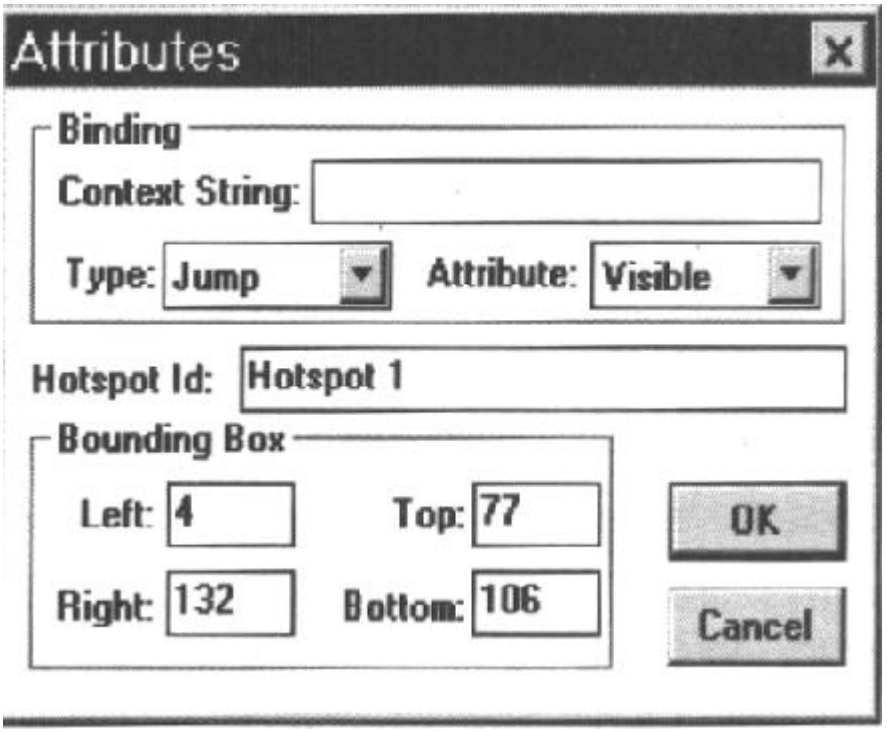


图 15.1　Hotspot Editor **不建立一个新图，只是在现有文件中建立热点**

注　属性对话框用于定义为图形所建热点的类型。

建立热点就像拖动鼠标一样简单：全部的工作就是建立一个如图 15.2 **所示的矩形框，它指示的区域就是热点。**

热点建完还需对它进行定义。双击热点，就会看到如下图所示的属性对话框：



其中只有三个需要指定的参数：内容字符串（Context String**），类型（**Type**）和属性（**Attribute**）。内容字符串的作用与前面介绍的双下划线标识的作用相同，是超链接的第二种形式，用户点击热点时就会转到相应的指定链接处。热点的边框可设为有或无，我一般都设为无边框。共有三种热点类型，下面分别予以叙述：**

技巧 使用 Edit|Preferences 命令可以定义一组标准的热点属性。该命令将打开一个对话框，在该对话框中定义缺省的内容字符串、类型以及属性。使用这个对话框还可以定义缺省的热点标识（ID）。

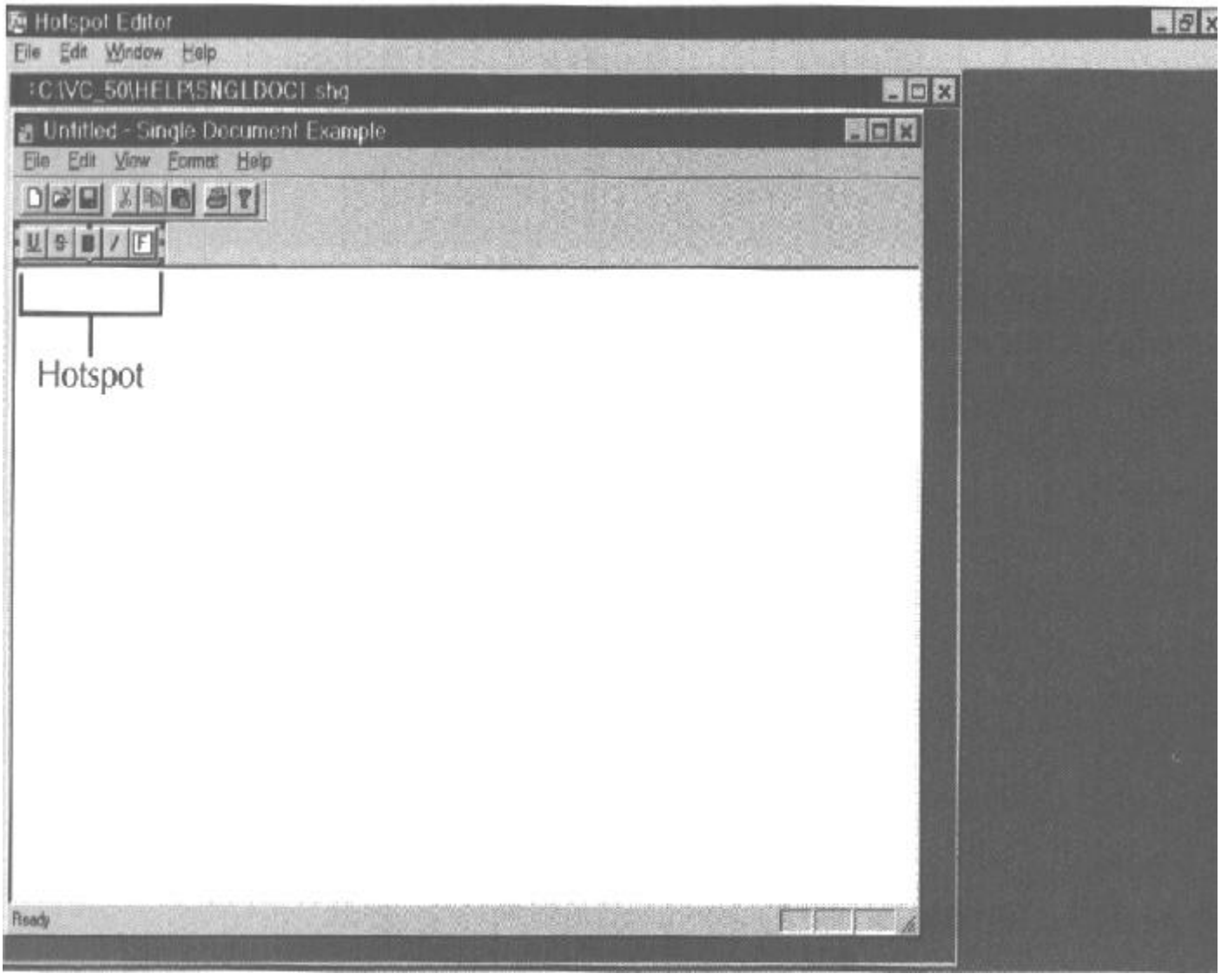


图 15.2 创建热点容易到只要拖曳下鼠标——方框指明了热点的出现位置

跳转型（Jump） 此时，无论内容字符串指示的帮助主题是什么，当前窗口的内容都会被替换，用户实际上从帮助文件的一个区域转到了另一个区域。适用于帮助用户查找同一主题下的多项内容，可以用它建立一个“参见”按钮；也可用于实现在多页结构图中的跳转链接。这种跳转是直接完成的，不需要再按“浏览”按钮。我还用它实现图形控件间的转移，以模拟实际的应用。

弹出型（Pop-up） 这类跳转最常用于控件描述或其它插图性的热点说明。与它相关的帮助主题显示在弹出窗口中，好处是用户无需离开原窗口，就可以方便地查看控件说明。我也将它用于教程型的帮助文件中的问答部分，按“回答”按钮，对问题的回答就会显示在弹出窗口中。

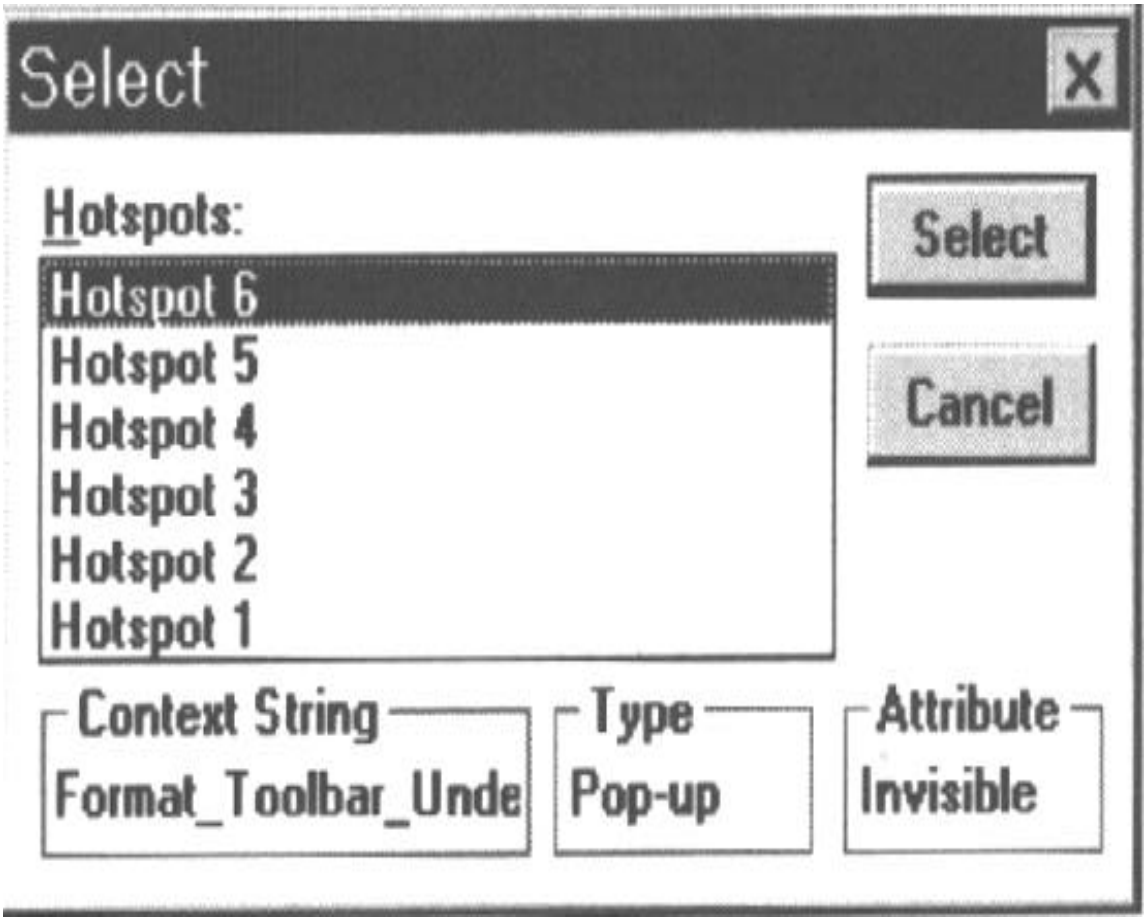
宏命令型（Macro） 可能有时不想让某个图形显示帮助文件中的内容，宏命令型的热点可以执行预定义的宏，后面会给出用宏建立 make 文件的例子，从中体会到 Windows 帮助中宏的强大功能，甚至可以用它改变 Windows 帮助的工作方式。

技巧 Microsoft Help Compiler 提供了许多预定义的宏。例如：SetPopupColor 宏命令改变弹出窗口的颜色。给用户设计一些按钮和菜单，实际是执行这些宏命令，会极大地提高帮助界面的友好性。用位图指示宏的方法是建立彩色方框位图，当用户点击该图时，帮助窗口的颜色会发生变化以匹配其内容。

属性对话框中还有一些有必要了解但无需更改的域。一个就是 Hotspot ID 域，实际是该热点的标识符，由于本书的读者多是程序设计人员，我后面会介

绍一下它的用法。另外四个域定义了热点的边框，用于调整边框的位置和大小，我觉得用它们来调整更方便更迅捷，用户只关心热点是否存在，而不会太在意位置是否偏离了一两个像素点。

有时你可能要重定义某个热点，对于那些可见热点当然好查找，但是我们说过，多数热点都定义为不可见的，那么怎么查找呢？用“编辑｜查找”命令。它的形式如下图所示：



注意这里显示出了当前图中的所有热点，是按热点标识符顺序列出的，而不是按内容字符串排序的。选中一个热点后相应的内容字符串就会显示，您就

可以验证是否是要找的热点，同时显示的还有跳转类型和属性信息，虽然微软没有加进范围信息，多数情况现有信息足够用了。

全部热点定义完后，可以用“文件｜保存”命令以当前文件名保存该图形文件，我一般都用‘文件｜另存为...’命令将它存成 SHG 文件，虽然 Hotspot Editor 没有坚持一定要这样做，但我建议最好还是做一下，否则您可能会发现某些更改过的信息用其它绘图软件读不出来。因为用 Hotspot Editor 不能修改图形本身，应该保持一份原文件的副本以便日后修改。

现在我们得到了图样（或其它多媒体元素），怎么加到帮助文件中呢？微软定义了一系列命令用以完成向帮助文件中加入图形或其它元素的功能。这些命令在一定程度上控制着这些控件的放置，但我觉得它提供的位置机制太原始了。表 15.2 是全部命令的清单。

注 一定要用“文件｜另存为...”命令保存编辑过的图形文件，否则原始图形文件会被覆盖，再想用就没有了。

技巧 可以在表 15.2 列出的三个图形命令前加上 T（透明），它将图形背景的颜色自动变为帮助窗口的底色，从而实现“透明”化。例如：{BMRT FIGURE.BMP}命令在屏幕右上角显示名为“figure”的位图，并将底色变为透明的。该特性仅限于 16 色图形。

表 15.2 多媒体帮助命令

命令	描述
{BMR <Filename>}	在窗口右侧显示图像。必须给出带扩展名的完整文件名。Windows 帮助系统识别位图文件(BMP、DIB、WMF)、多热点位图文件（SHG）和多解析度位图文件（MRB）。但不幸的是，不能使用 PCX 文件
{BMC <Filename>}	在窗口中央显示图像
{BML <Filename>}	在窗口左侧显示图像
{MCI_LEFT [<Options>,] <Filename>}	在窗口左侧显示多媒体控件（MCI 文件）。在 Microsoft Help Workshop 帮助文件中有一个错误，它说该选项只支持 AVI 文件，其实最新版的 Windows 帮助系统支持所有的 MCI 格式，包括 WAV、MID 和 AVI。最好只用这三种格式文件，除非您知道所用格式是现用系统支持的。可以给出一或多个选项，包括：EXTERNAL、NOPLAYER、NOMENU、REPEAT 和 PLAY。 EXTERNAL 选项保证该文件独立于帮助文件，从而减少了加载帮助时占用的空间；不便之处在于多媒体文件必须作为帮助中的独立项目。Windows 帮助在显示文件时总是显示相关的媒体播放器，可以用 NOPLAYER 选项屏蔽它，实现自动播放和重复播放的功能。NOMENU 选项显示一个不带菜单按钮的播放条，它保持了播放条的基本功能但不能重放。REPEAT 选项则实现第一次播放完成后立即重播功能。PLAY 选项进行自动播放——非常适用于闪烁窗口

续表

{MCI_RIGHT [<Options>,]<Filename>}	在窗口右侧显示多媒体控件（MCI 文件）

技巧 为支持多种显示方式，可以为同一命令指定多个位图文件。Windows 帮助系统自动选用最适合的位图文件。例如：命令{BMR CAT016.BMP;CAT256.BMP; CAT024.BMP}分别提供了 16 位、256 位和 24 位三种彩色位图供不同的机器使用。更进一步，还可以用 MRB 文件来提高帮助文件的灵活性，其好处是：您能够非常灵活地显示详细信息，当然前提是用户的机器支持 WMF；坏处是：帮助文件骤然增长，大吃内存。

注释 进一步扩展帮助文件的帮助语句是：{EWx<DLLName>, <WindowName>, <Data>}，可以用它访问外部的 DLL，其中 EWx 的 x 指示 DLL 中左（l）、右（r）和中央（c）的位置。WindowName 包含当前帮助文件窗口的名字，前面我们说过是用$脚注标出的。Data 用于向 DLL 中传递待处理的数据。

注 要慎用图像和声音文件，以防帮助文件过长。

帮助文件中肯定可以有响铃、鸣叫以及其它应用程序能具有的各种多媒体效果。但是你应该仔细考虑，不要因为追求这些效果而带来很大的负面效应，

即使很小的多媒体程序也要占较多的资源，所以只有当确信它们能带来理想的效果时才使用图像和声音。

编写帮助文件时还必须考虑内存的开销。当用户使用帮助时，Windows 帮助系统总是装载所有文件。减少占用内存量的一种方法是使用许多外部文件，即将帮助文件分成许多部分，多媒体都存为外部形式。少量使用图像和声音的目的之一是使你的帮助文件增加趣味性。

建立 Make 文件及编译帮助文件

建立一系列的 RTF 文件可能是编写帮助文件中最费时的工作，与编程时所用的 make 文件（工程文件）不同，编译帮助文件的 make 文件含有的不只是要编译的文件的清单。

本节我们将介绍手工建立 make 文件的方法，Microsoft Help Workshop 程序提供了一种自动的方法我们将在随后的章节中介绍。先介绍手工方法的原因是，使用 HC31——DOS 版的编译程序时必须用它，而且清楚了 make 文件的组成有助于以后调整其结构。

我们从观察一个典型的 make 文件开始。程序列表 15.2 包含了本书中使用的帮助文件的 make 文件。如果用 DOS 程序来编写帮助文件，所有的工作都必须手工填写；不同于我们所看到的 make 文件顶部的注释中的说明，该说明声称帮助编译器自已维护文件，不要开发人员直接修改它们。后面我们将用 Microsoft Help Workshop 程序建立同样的帮助文件。这里我所用到的一些东西并不是典型帮助文件所必须具备的，我会进一步讲解何时用到它们。

注释　程序列表 15.2 中的有些条目因为过长可能被分写成了数行，真正使用这个程序列表时一定要保证每个条目都在一行上。

程序列表 15.2

```
;This file is maintained by HCW.Do not modify this file directly.

[OPTIONS]
HCW=0
COMPRESS=12    Hall Zeck
ERRORLOG=HELP.LOG
LCID=0x409    0x0 0x0 ;English (United States)
REPORT=Yes
CONTENTS=CONTENTS
TITLE=Single Document Application Help File Example
COPYRIGHT=1998    Some  Company
HLP=.\Sngl_Doc.hlp

[FILES]
.\Sample.rtf
.\Glossary.rtf
```

```
[MAP]
#include ..\Resource\my.hm
CONTENTS=1          ;Main help file menu.
FORMAT_TOOLBAR=2    ;Format toolbar explanation shortcut.
GLOSSARY=3          ;Glossary window shortcut.
HID_FORMAT_FONT=0x1E160    ;Added for ID_FORMAT_FONT in AFXRes.h file.

[WINDOWS]
Main="A Sample Help File",(0,0,800,600),52484,(rl4876671),(r12632256),f7;This is the main window.
Glossary="A Sample Help File - Glossary",(50,50,850,650),52848,(rl4876671),(r12632256),f7; The direct Glossary help window.

[CONFIG]
CB("glossary","&Glossary","JI('SNGL_DOC.HLP>glossary','GLOSSARY' )")    ;Add a Glossary  button to the display.
CB("controls","C&ontrols","JI('SNGL_DOC.HLP>main','FORMAT_TOOLBAR')")    ;; Add a jump to the controls bitmap button.
RegisterRoutine("USER","EnableMenuItem","uuu")    ;Enable (or disable) a menu item.
RegisterRoutine("USER","GetSubMenu","u=uu")    ;Get the name of a submenu.
```

```
RegisterRoutine("USER","GetMenu","u=u")      ;Get a menu name.
RegisterRoutine("USER","GetActiveWindow","u=")      ;Get the active window name.
RegisterRoutine("USER","DrawMenuBar","u")      ;Instruct Windows Help to draw a menu bar.
EnableMenuItem(GetSubMenu(GetMenu(GetActiveWindow()),1),0,1027)      ;Disable the copy option of the Edit Menu.
DrawMenuBar(GetActiveWindow())      ;Redraw the menu when we're through.
```

正如你所看到的，这个 make **文件看上去有些庞大，我们最好分节来解释。例如，观察** FILES **一节，它列出的是脚本文件（按微软的叫法是“主题”）清单，我们先仔细讲解一下第一小节：**

Make **文件的** OPTIONS **小节指示帮助编译器如何编译文件。这里的大部分条目都是不言自明的（当然是对英文好的人士来说——译者注）。例如：** COPYRIGHT **语句出现在帮助窗口的“关于...”对话框，**COMPRESS **语句指示编译器是否对文件进行压缩以及如何压缩，**HCW **语句是** Microsoft Help Workshop **专有的，使用** DOS **编译器时用不到。最需要注意的是** HLP **条目，帮助文件的文件名和** Visual C++**程序名相同，虽然这不是必须的要求，但使用这样的名称却会极大地简化以后的操作，**MFC **会自动完成调用与程序同名的帮助，当然二者的扩展名是不同的。**

MAP **小节用于设置帮助内容的属性，在此出现的每个词都必须是脚本文件中的某个帮助主题标识符（#脚注所标出的内容）。为每个跳转赋值实现了从控件中访问帮助主题的功能。用户只需选中控件按** F1 **就可以看到关于该控件的帮**

助。注意：我们没有把程序中用到的所有控件都罗列出来，而只是包括了 HM 文件，其好处是不仅节省了录入时间，还有助于发现帮助文件中的漏洞。

有多种方法显示帮助文件中的数据，我一般为每个主题开设一个窗口。在 make 文件的 WINDOWS 小节列出了两个窗口：一个是主窗口，这个窗口总是显示的；另一个是词汇表窗口，当用户点击工具栏上的“ Glossary ”按钮时就会显示该窗口，通过它用户可以在不失去当前位置的前提下查找单词。显然并不是每次用户访问词汇表时都显示这个窗口，如果用户点击帮助文件中的超链接文本访问词汇表，词汇表就显示在主窗口中而不会另开一个窗口，这时用户只需用“ 返回 ”按钮就可以返回到以前的位置。

CONFIG 一节是我要着重介绍的，这是建立 make 文件时费时最多的部分，原因也正在于它最灵活。我的 make 文件中有三类事件：建立按钮、注册函数及执行一系列的 Windows API 调用。显然，可以加入任何事件，我们先看一看下面这个相对简单的例子：

由建立两个新按钮开始。第一个是“ 词汇表 ”按钮，这时要用到两个不同的宏调用，一个用于建立按钮，以“ Glossary ”作为它的标签名并在字母“ G ”下加下划线，功能是跳转到 JI 宏调用所返回的标识符位置，JI 宏调用看上去使用了两个参数，实际使用了三个参数：“ SNGL_DOC.HLP>glossary ”实际提供了两条信息：一是告诉 JI 宏使用哪个帮助文件，二是告诉 JI 宏使用哪个窗口显示帮助主题。另一个参数提供主题标识符的名称——请记住，这是一个#型脚注。

接下来要做的工作是向 Windows Help 注册一些 DLL 函数。注册的函数数目不受限制，注册过程分为三步：第一，告诉 Windows Help 调用哪个 DLL，

Windows Help 总是假设 DLL 在 SYSTEM 目录下，所以要么把 DLL 放在这个目录下，要么给出它所在的路径。我一般都是放在 SYSTEM 目录下，因为无法预测用户将来会把它放在哪里。第二，给出该 DLL 中所用的函数名字，注意大小写要与原函数完全一致——过去我曾遇到过因函数名不一致带来的麻烦。第三，告诉 Windows Help 该函数将要查找何种参数。当我注册 GetSubMenu 函数时需使用“u=uu”，即：此函数需要两个无符号数作为输入，返回一个无符号数作输出。共有四种数据类型：无符号数（u），带符号数（i），字符串（s）或未知类型（v）。等号（=）总是用于分隔输入值与输出值。

在注册过 API 调用之后，可以使帮助的显示发生一些变化（可以做的包括所有注册过的函数提供的功能）。以禁止使用 Edit|Copy 命令为例，虽然还可以看到它，但已变为灰色的，即不可执行。使用这种技术可以使帮助做的非常时髦，注意完成时要用 DrawMenuBar 函数来重画菜单条，否则所做的修改可能显示不出来。

最后一步就是编译帮助文件了。如果用的是 DOS 版的帮助编译器（HC31.EXE），必须在[CONFIG]小节给出记录文件（log 文件），我的 make 文件中用 ERRORLOG=HELP.LOG 语句来实现，使我可以用文本文件来查看帮助文件中的错误。不便的是，DOS 版的帮助编译器是命令行驱动的，界面不够友好，如果不指定记录文件，你就只好期望有一目十行的本事了。所要做的很简单，键入：HC31<MAKEFILE>.HPJ，make 文件一定要带 HPJ 扩展名，否则编译器找不到。在建立过程中，编译器会在屏幕上显示许多“……”，当再次显示 DOS 提示符时，表示编译已经完成了。

15.4 使用 Microsoft Help Workshop

早期的 Windows 程序设计人员即使编译一个很小的工程文件也要学习很多东西，那时的工具很简陋，即使一个小程序也要编很多代码。那时是 DOS 的天下，但随着编译程序的改进，DOS 已是昔日黄花；但编译帮助文件时还是要用到那些编译程序。实际上我也很反感用 DOS 程序来建立帮助文件，但是在最新版的 Windows95/98 编译程序出现以前也只能使用它。即使是第三方厂家专门开发的编译程序有时也要和晦涩的 HC31 命令行程序打交道，它们在编译过程的可视性方面并没有实质性的改进。

注释 本节专门讨论了 Windows 帮助文件，如果关心 HTML 帮助文件的建立技术，请阅读后续章节。

在引入了 Microsoft Help Workshop 之后，编译过程有了全面的改观。再也不用依赖于基于 DOS 的编译程序了，Microsoft Help Workshop 在 Windows 环境下建立并编译帮助工程文件（帮助 make 文件的另一种叫法）。图 15.3 是 Microsoft Help Workshop 典型的显示示例。

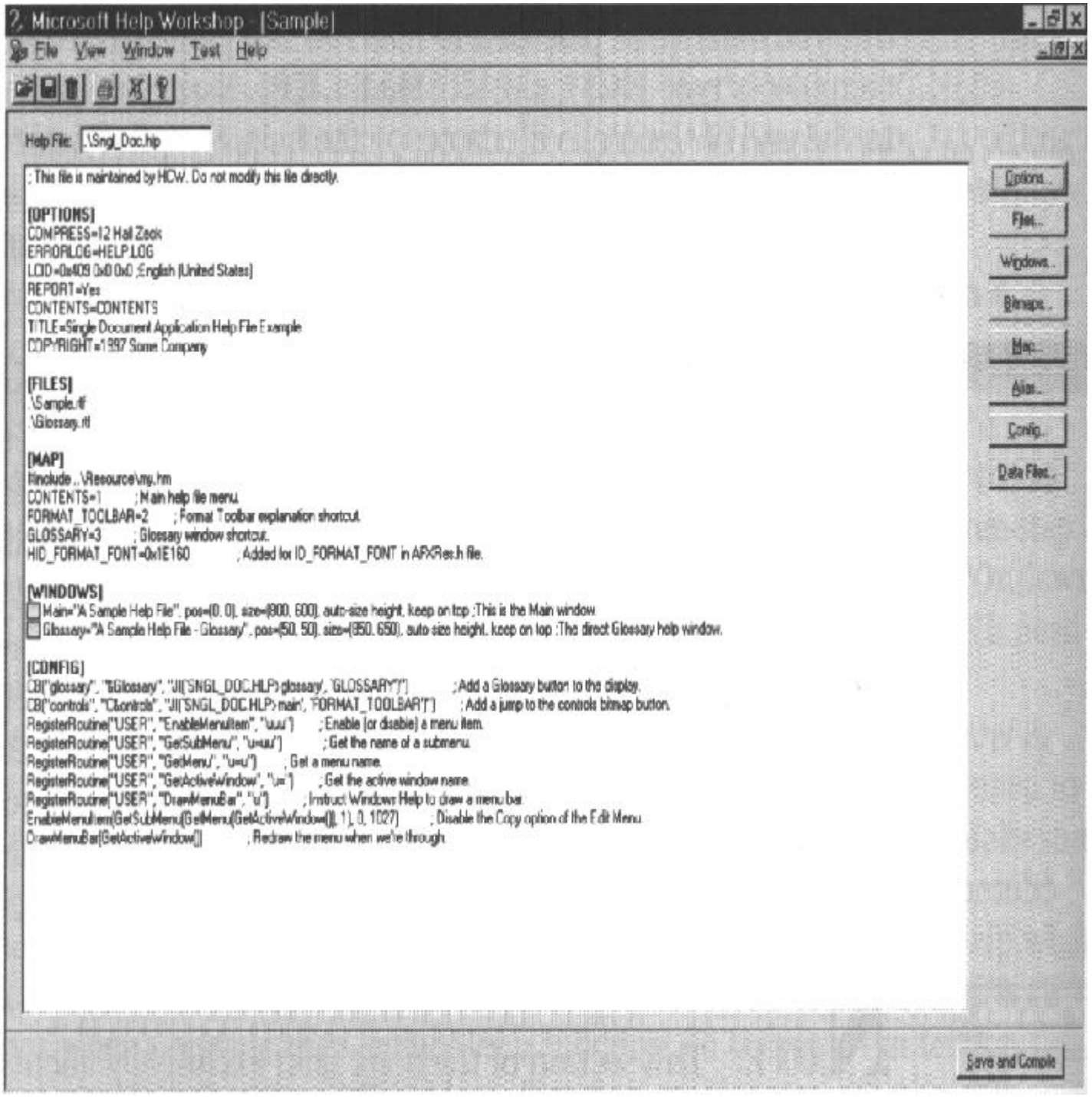
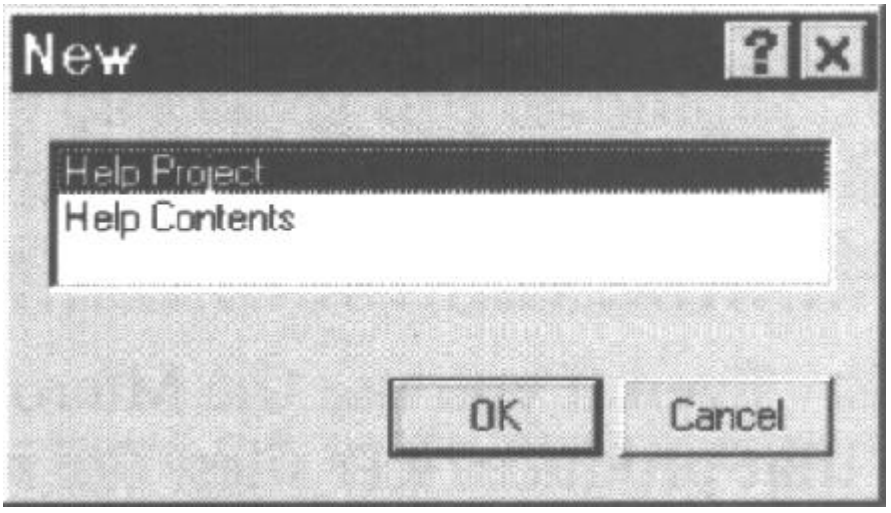


图 15.3 Microsoft Help Workshop **提供了一个方便的** GUI **用以建立帮助文件**

注释 即使使用 Microsoft Help Workshop，也还是需要一个生成 RTF 文件的编辑器。我不知道 Microsoft 为什么没有改进这一方面，事实就是如此。我还是用 Microsoft Word 建立 RTF 文件，在使用 Microsoft Help Workshop 建立帮助文件时，所用的 make 文件指向这些文件。

我们看看这个工具如何降低建立过程的复杂性，所建的 make 文件如程序列表 15.2 所示（假定已经建立了包含帮助脚本的 RTF 文件）。第一步是建立一个新的工程文件，用 File|New 命令实现，系统会显示如下图所示的对话框：

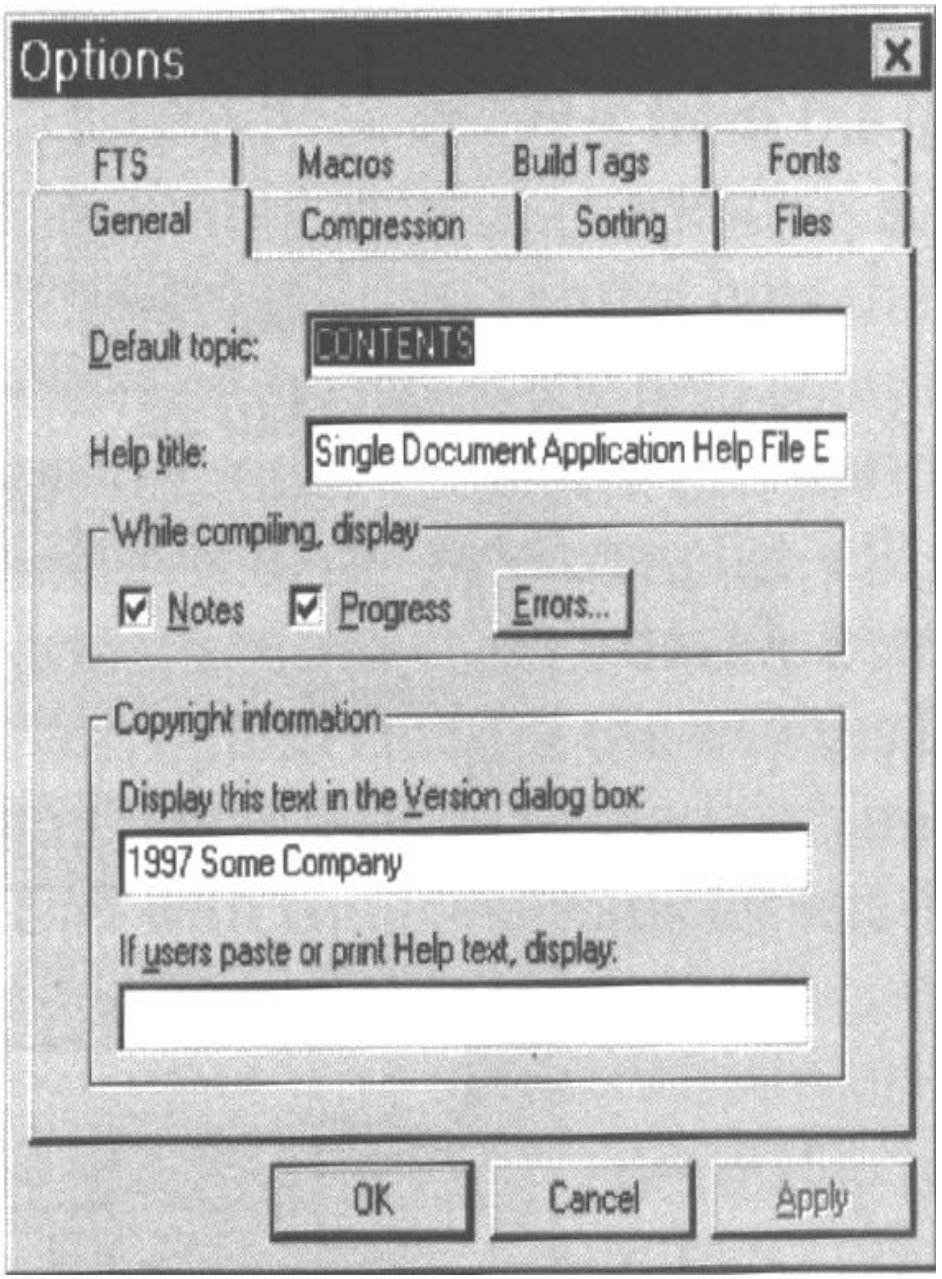


选择 Help Project 后，按 OK 完成。如图 15.3 所示，已经建成了一个新的帮助工程文件，它初始为一个空白文件，由你填入帮助文件的特性。

警告 用 Microsoft Help Workshop 建立的文件与用 DOS 编译程序建立的文件互不兼容，必须预先做好决定：新的 Microsoft Help Workshop 对开发者更好用，但只限于 32 位的操作系统环境；如果还需要对 Windows 3.x 向下兼容，就只好用 DOS 编译程序了。

定义工程的各种选项

当开始使用一个新的工程文件时，总要先定义一些选项。例如：至少要知道调用帮助文件的目的，要加进哪些版权信息。我一般以内容主题为主目录，所以把它加在帮助文件的开头位置是适宜的。点击 Options 按钮，系统就会显示 Options 对话框：

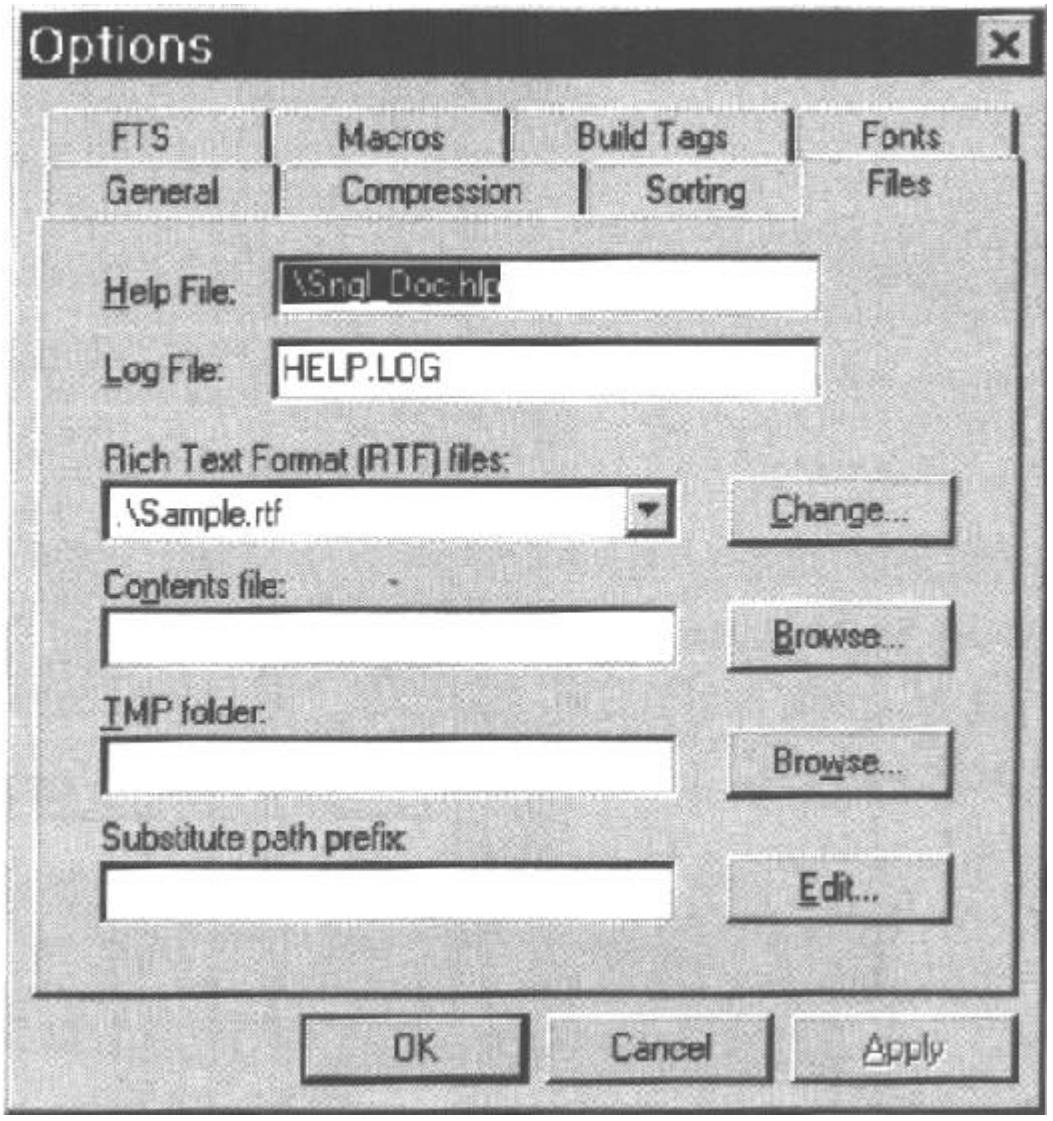


注意此时已经定义了一些基本选项，这些选项的含义类似于前面章节中 make **文件描述的内容（你会看到** Compression **属性页上的选项相当熟悉，它们也是用来压缩帮助文件大小的）。此时，有些人会觉得这个工具并没有什么优越性，特别是当他们的盘上有很多预定义文件时。要指出的是这个对话框使你只需填**

填表，而不用再一条一条地完成语句。

Options 对话框的“ Sorting ”属性页有二个区域，第一个区域决定了帮助文件使用的语言，语言决定了排序原则，每种语言的字母顺序都不相同。第二个区域有两个选项，一个选项允许忽略非空格字符，例如：如果不选此项，出现在 ê 上的（ ˆ ）就会影响排序；另一个选项表示排序过程忽略帮助文件中的所有符号项，这对于为数据录入程序或其它一般应用程序建立非特殊的索引时非常方便。另一方面，为参考手册型的帮助文件建立索引时它又确实很碍事，因为许多的 C 函数都是以“ _ ”开头的，忽略“ _ ”就不容易识别函数。

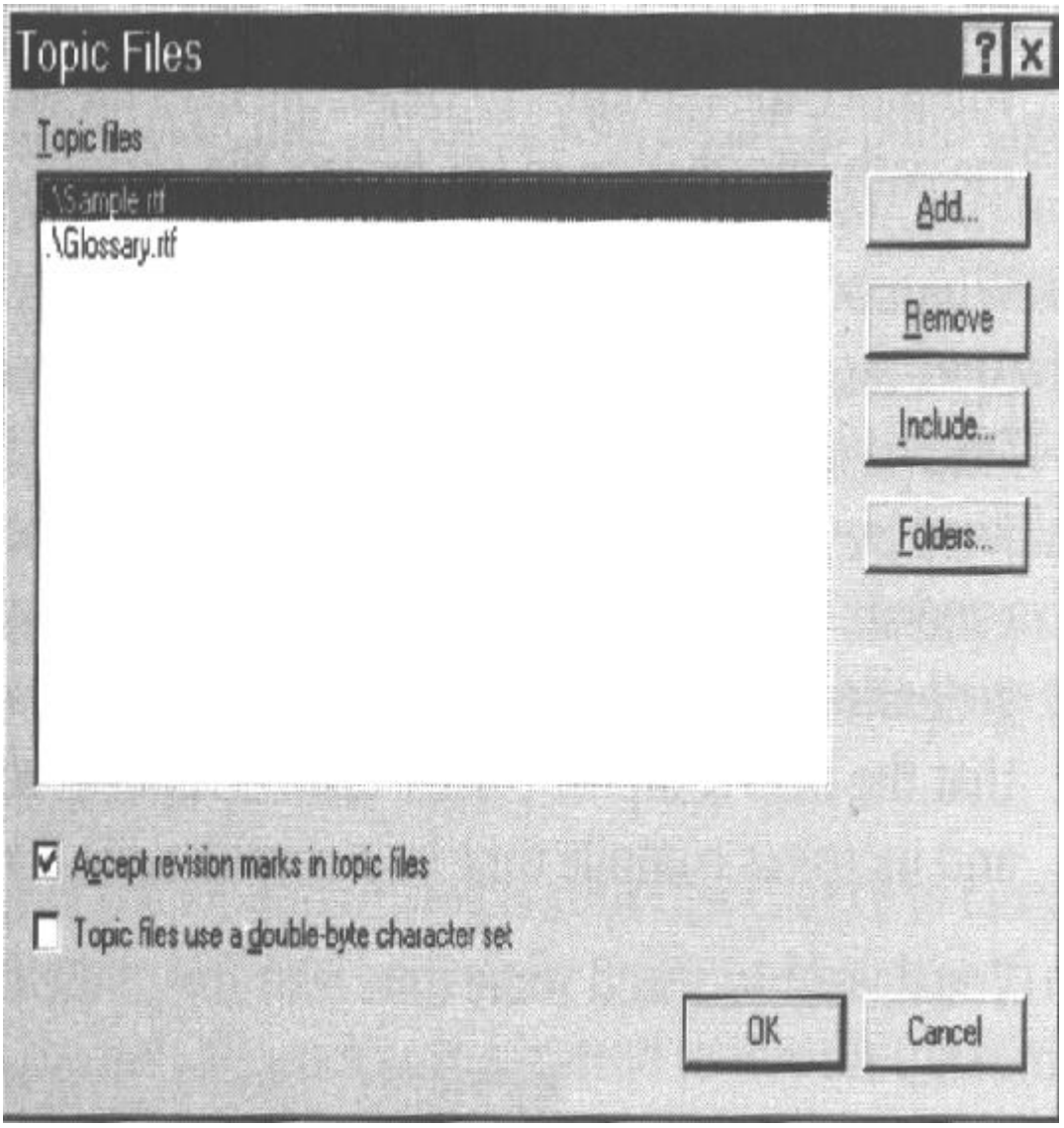
Options 对话框的“ Files ”属性页如下图所示：



你会注意到它的许多选项我们以前都介绍过，修改 Help File **域可以给帮助文件改名，一般帮助编译器用工程文件名作为帮助文件名的基础；**Log File **域包含了记录文件的文件名，所幸的是这一项在新的帮助编译器中已不是必需的了。我还是用它来记录编译过程，但现在这是可做可不做的事情。**

在此属性页上的最重要的域是 RTF **文件清单框，当前帮助文件的工程文件**

所含的文件都列在这里，点击“Change”按钮显示“Topic Files”对话框，如下图所示：



从这里可以增加或删除工程文件中 FILES 小节所列出的主题文件。注意此对话框中的两个复选框，它们很重要，因为它们控制着帮助编译器如何响应 RTF 文件。第一个选项允许编译器在下一次编译时自动执行对 RTF 所做的更改，如

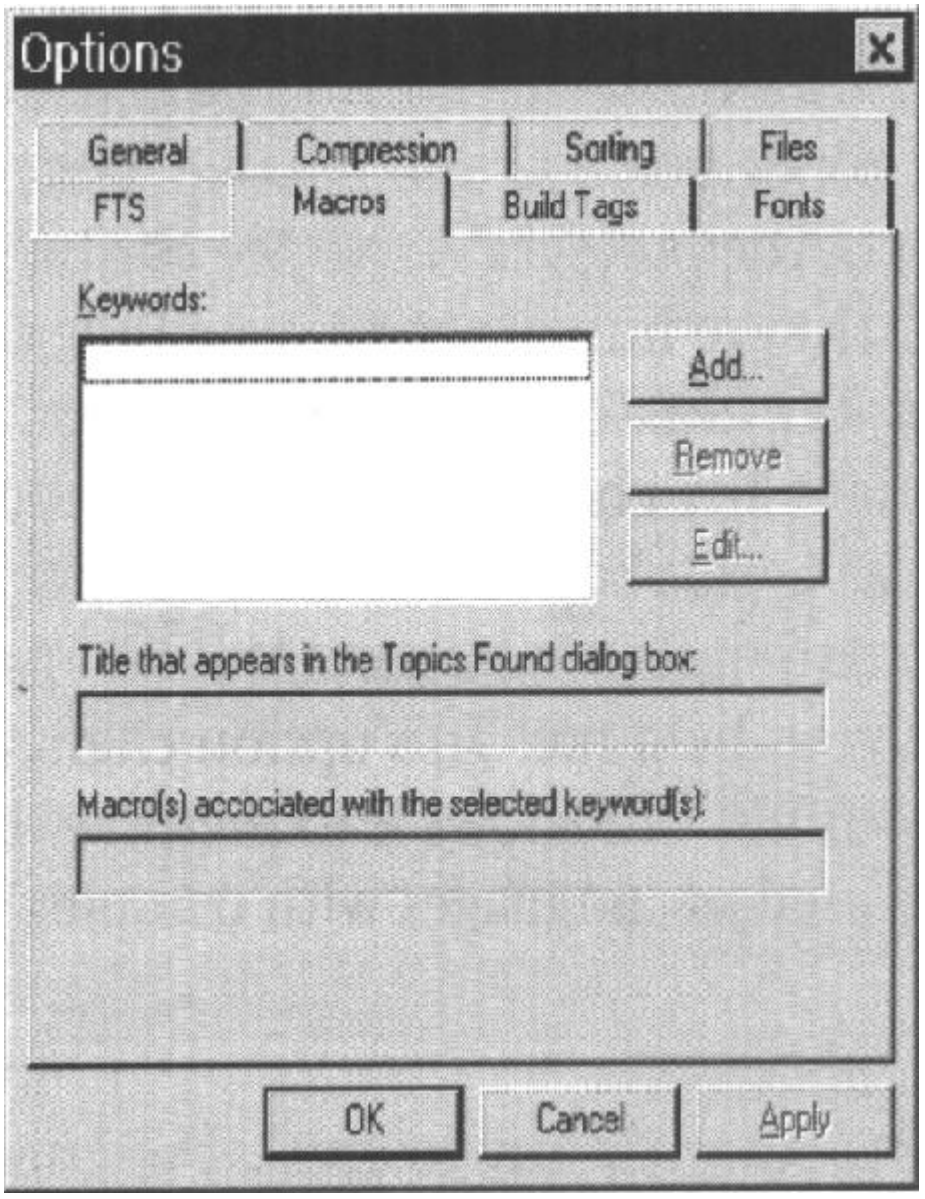
果未选中该项，编译器就会忽略所做的改动。第二个选项对使用双字节字符集（DBCS）的帮助文件非常重要，它改变了帮助编译器的工作方式，从而保留了特殊字符（该特征主要用于特殊的语言，比如中文等）。

技巧 进入“Topic Files”对话框的另一途径是单击主窗口的“files”按钮。

在‘Options’菜单的‘Files’属性页还有其它一些选项，其中之一就是‘Contents File”域，如果是新建一个工程文件，Help Workshop 在建立内容页时就会自动地填充该项。选中该选项的原因是已经有了一个内容页又想用于当前的工程文件中。当帮助文件超过 8MB 时要用到 TMP Folder 选项，它给出了 Help Workshop 编译时建立临时文件的目录，一般情况下用不到，只是当缺省目录的磁盘空间不足时才用到。最后一个域是“Substitute Path Prefix”，当移动了某些文件又不想改变工程文件中的路径信息时要用到。

Windows 95/98 的帮助文件提供了一些以前没有的功能：全文查找。当选中“帮助主题”对话框的“查找”页就会建立这个数据库，实现对全文的逐字查找。编译过程中“选项”对话框的 FTS 页中有一个选项用于生成这个文件，由于 Windows 95/98 总是要生成它，一般都不用选中。编译器建立的 GID 文件要在发行盘上占用很多空间并增加了编译时间。

下面有必要学习一下“Macros”属性页，其图像如下图所示：



从这里定义关键字宏供全文使用。而且当用户查找某一主题时，这些宏会出现在帮助主题对话框的索引页中。

单击“ Macros **”属性页的“** Add **”按钮，系统显示一个含有三个域的“** Keyword Macors **” 对话框，第一个域含有宏名字，第二个域指示宏本身，第三个域指示**

Help Workshop 如何在索引页显示该宏，它用于有多个帮助文件但只想在某个选定文件范围显示关键字的情况。例如：一般把词汇表和程序列表名的缩写集中存放在一个独立的文件中，用 JI 宏建立一个全文范围的跳转。就用关键字宏的方法来实现，用户根本不会感觉到又加载了其它的文件——因而对用户是透明的。

注 请谨记，Options 对话框的 Macros 属性页的设置会影响整个帮助文件，而不只是一个窗口。

前面我们介绍过“*”脚注用于建立标识，Options 对话框的 Build Tags 属性页就该用到它了，前面对这个脚注已经讲得十分透彻，这里不再赘述。中心思想就是给 Help Workshop 提供帮助文件要包括的建立标识的一个清单。即使 RTF 文件含有主题，如果不做主题标识，该主题也不能出现在帮助文件中。如果没有填写该属性页，Help Workshop 会包括所有 RTF 文件中的所有主题。

Options 对话框的 Fonts 页是第一个用于定制帮助文件外观的地方。我一般不用，而是在字处理过程中就完成了。然而，如果用文本编辑器建立了一个 RTF 文件，再用该选项可以节省时间。Character Set 域允许选择字符集，缺省为 ANSI，可以选择其它的语言，如阿拉伯语。在 WinHelp Dialog Boxes 域的 Font 用于指定缺省字体，单击 Change 按钮，显示含有三个域的 Font 对话框。第一个定义字体名称，第二个定义字体大小，第三个定义字符集。在 WinHelp Dialog Boxes 域的 Font 下拉列表框用于更改 Windows 帮助文件的通用字体，允许用一种字体代替另一种字体，Add 按钮显示一个 Add/Edit Font Mapping 对话框，它含有两个组，每个组有三个域。这三个域与前面描述的 Font 对话框中的相应域完全

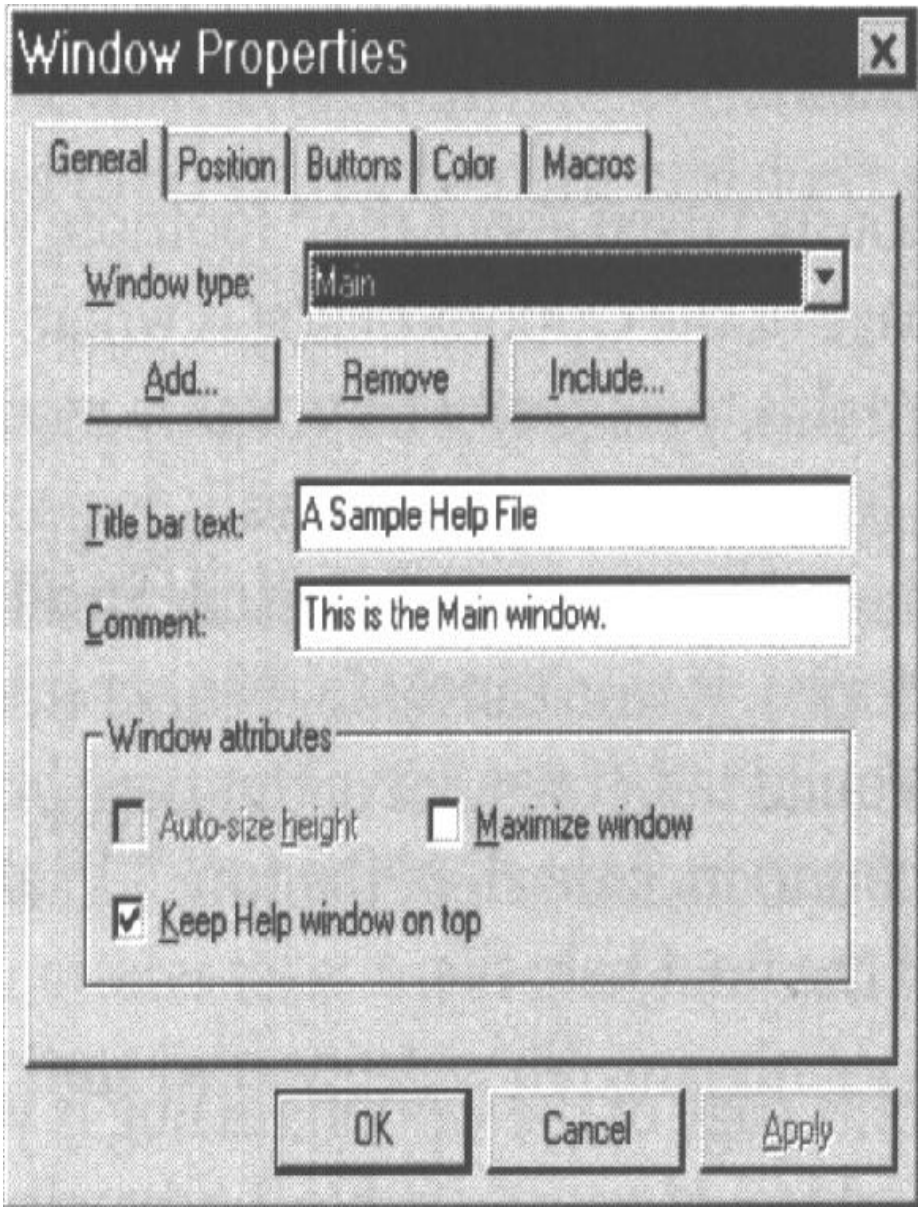
一致，唯一的问题是如果字处理程序覆盖了这些设置，那么它们就不起作用了——这是使用像 Word for Windows 这样的产品编辑文件时经常发生的事。

定义窗口

定义窗口选项只是建立工程文件的第一个步骤。当这些选项起作用时，需要定义一些窗口来显示数据。一般建立一个叫 Main 的窗口，它是应用程序用到的主窗口。

建立窗口很简单，只需单击 Main 窗口的 Windows 按钮（图 15.3），系统就会显示窗口属性对话框：

首先应关注的是 General 属性页。单击 Add 按钮，系统显示一个带有两个域的 Add a New Window 对话框，一个域是窗口的名称，另一个域指明窗口类型，Help Workshop 可以建立三种类型窗口：过程窗口、引用窗口以及出错消息窗口。过程窗口和引用窗口的差异很小，它们都可以自动调整大小，并且都包含了三个系统按钮。这两类窗口的最大差别是它们在屏幕上的放置位置——后面我们将会讲到如何设置位置信息。出错消息窗口不同于其它两个窗口，它并不包含三个系统按钮，而更像一个对话框。

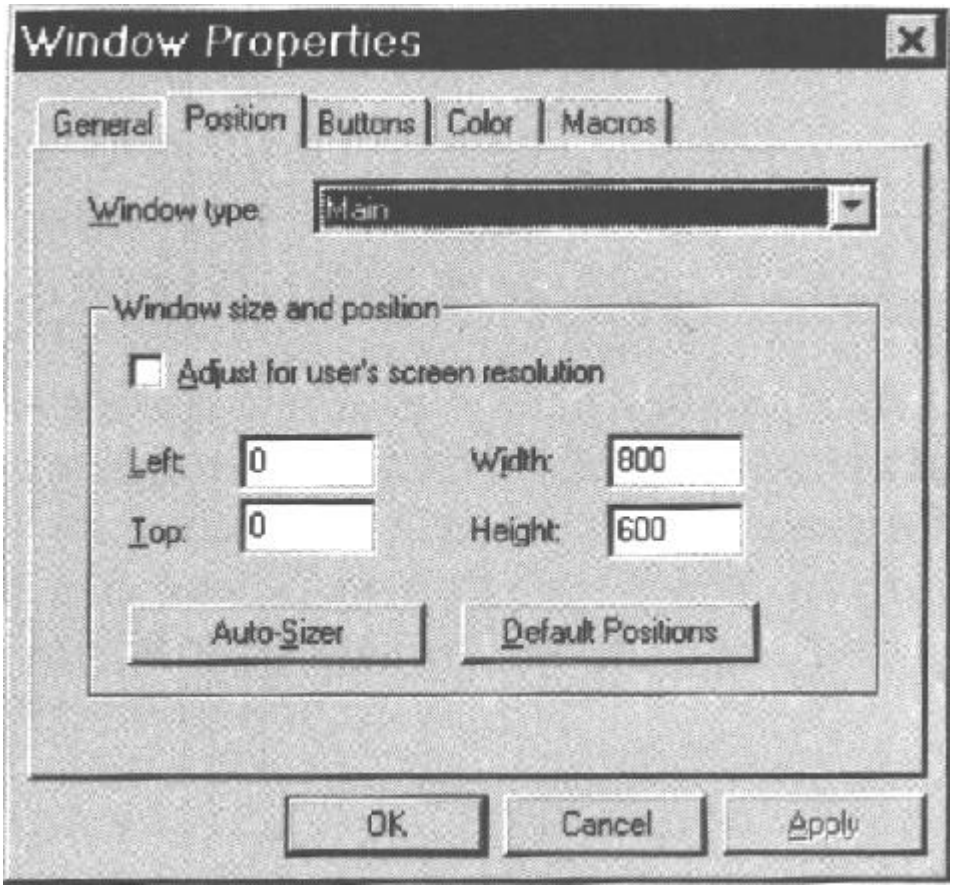


返回到 General **页，**Title Bar Text **域定义了** Windows Help **在标题栏里放置什么，它不影响帮助窗口的标题区的实际显示。**Comment **域是在工程文件中加入注释，我总要在工程文件中加入注释，以帮助随后理解帮助文件的建立过程。该属性页中还有三个属性复选框，**Help Workshop **可以根据当前情况禁止其中一**

个或多个复选框，例如：你不能让主窗口自动调整大小，否则当允许某个辅助窗口自动调整大小时，该窗口打开后，用户就无法使其最大化。大部分过程窗口缺省情况下总显示在屏幕的最外层，如果你想让操作你的程序的用户随时都可以看到帮助，这就是实现该功能的一个便利的特性。

技巧 为了更好地控制窗口显示的外观，我总是关闭 Auto-Size Height 特性。Position 属性页的选项将对帮助窗口提供全面的控制，后面我们会具体介绍该属性页。

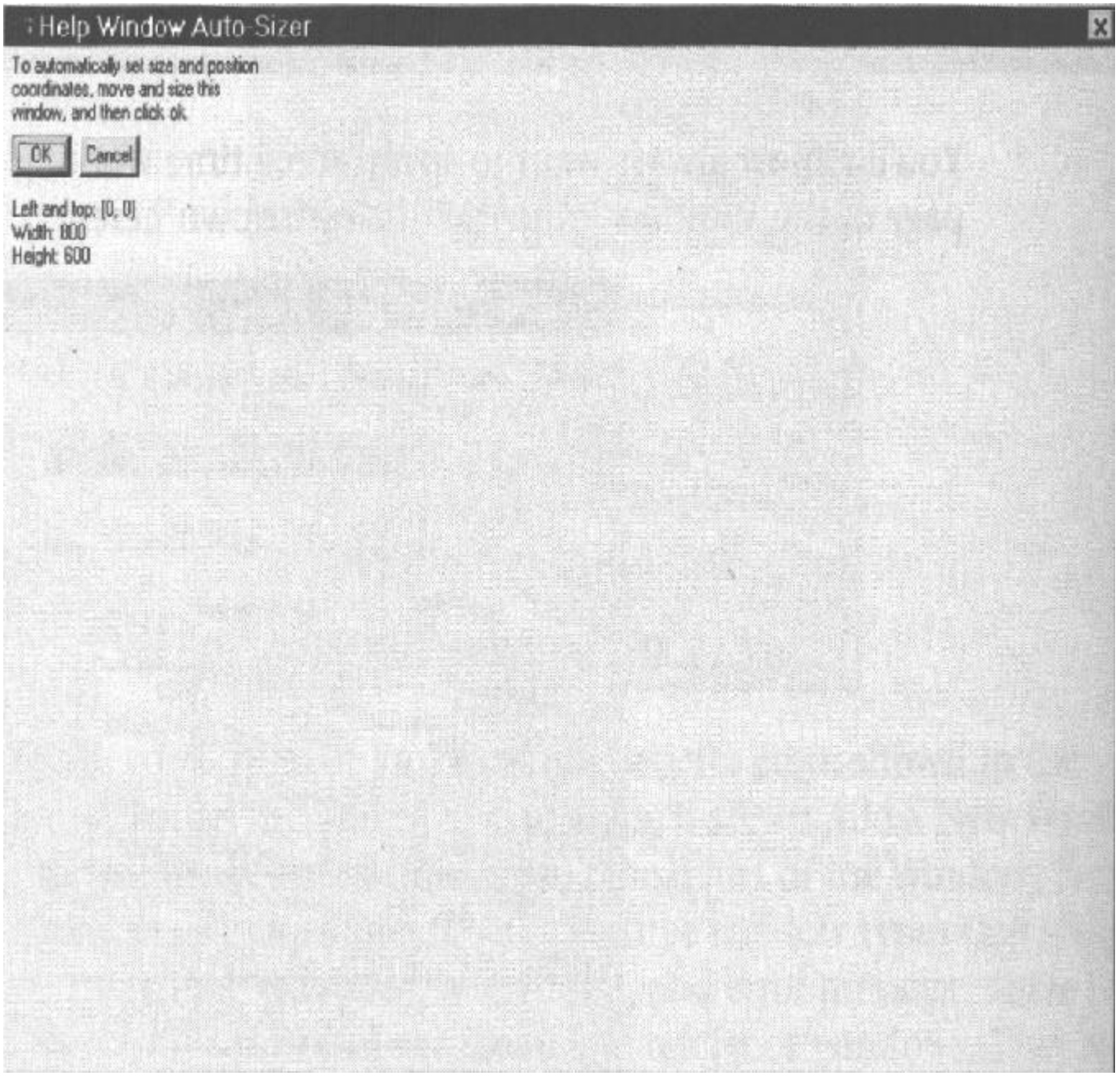
你可能会经常调整 Windows Properties 对话框的 Position 属性页，该页如下图所示：



它比其名称所包含的功能要强一些，该属性页除了能控制帮助窗口的位置及大小外，还提供了很多非常容易使用的特性。

在这个属性页上有四个域：Left、Top、Width **和** Height**，它们控制窗口的大小和位置。一般第一个帮助窗口放在左上角，大小为** 640×480 **或** 800×600**，具体使用哪一个值要随机器的性能而定。这样的窗口看上去可能会小一点，但用户一般都会自行调整。如果程序员将窗口放得离边缘太近，使用时要想在用过新窗口后又返回老窗口时就很不方便。好在还有** User's Screen Resolution **选项，当用户的显示器档次太低时，用该选项可以防止帮助窗口被全部隐藏起来。**

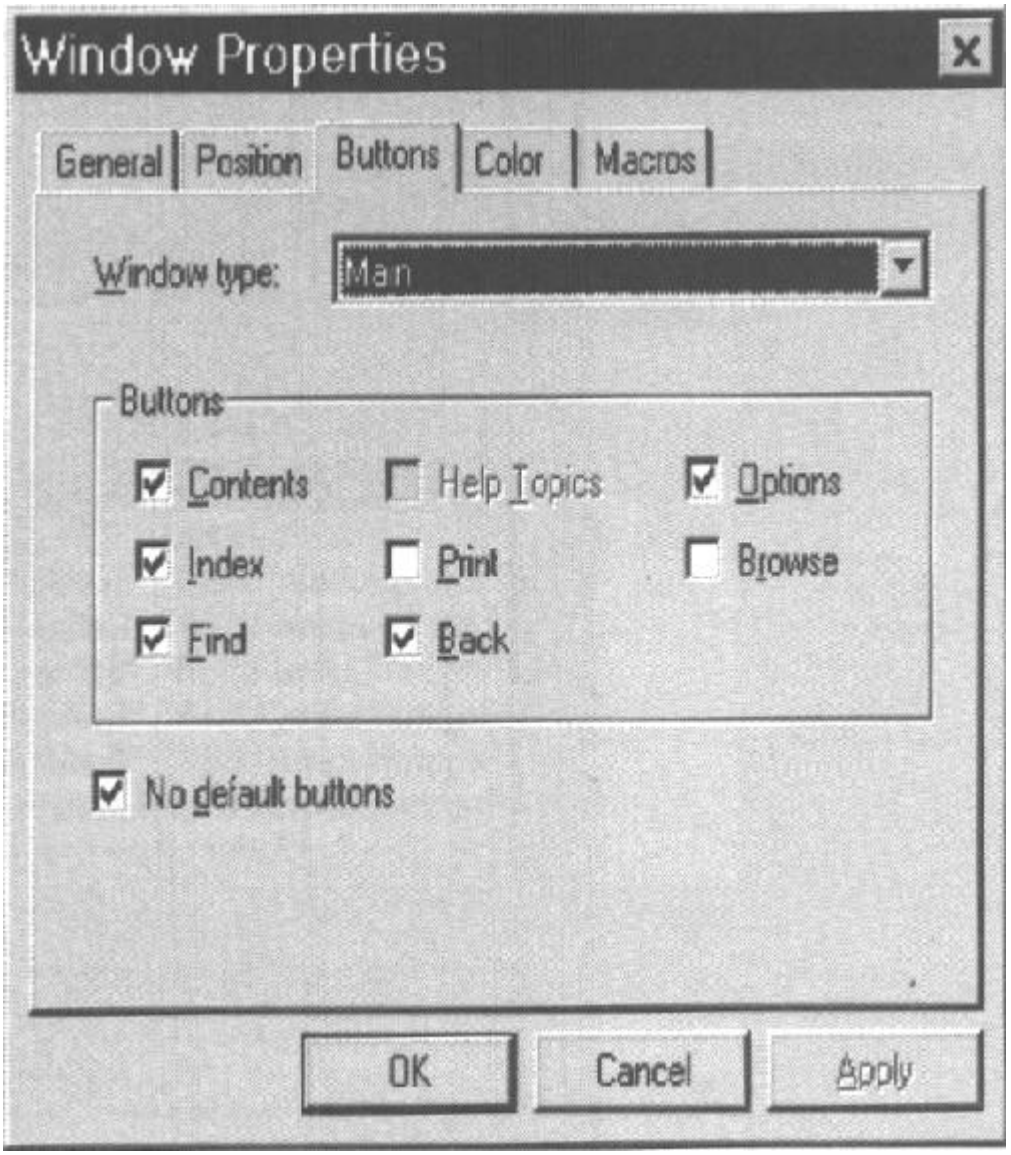
该属性页还有一个非常特别之处，起初你可能会不注意，看一下 Auto-Size 按钮，单击它会显示如下图所示的窗口：



如果改动了窗口的位置，Left 和 Top 域的值也随之而变；重新改变窗口的

大小则会改变 Width 和 Height 域的值。这种用图像方式改变窗口的方法减少了重编译帮助文件的次数。

Windows 95/98 定义了许多可以加入帮助文件的缺省按钮，有时可能不会都用到，例如：如果在 RTF 文件中未定义浏览（+脚注）功能，那么 Browser 按钮就用不到了。这里显示的 Buttons 页用于定义帮助窗口中用到的按钮。



所有辅助的过程窗口和引用窗口都用不到 Contents 和 Index 按钮，主窗口中则把它和 Print 及 Back 按钮一起作为缺省按钮提供；另一方面，主窗口不用 Help Topics 按钮。与所有其它类型的窗口不同，出错信息窗口没有任何限制，可以随心所欲地使用各种按钮。

技巧 单击 No Default Buttons 复选框，可以不受 Help Workshop 在主帮助窗口按钮设置方面的限制。这个复选框仅在主窗口中出现，即不能修改辅助性过程窗口和引用窗口中存在的限制。

下面介绍 Color 属性页，它有两个域：Nonscrolling Area Color 和 Topic Area Color。每个域都有一个 Change 按钮。单击它会出现一个调色板，从中选择不同的颜色，帮助窗口的颜色也会随之而变。

最后一个属性页是 Macros 属性页，它的外观如下图所示：



主窗口总是使用工程文件中 CONFIG **节的宏作为缺省宏，本节中你看到的所有宏都是可自我执行的——这就是** CONFIG **节的宏都加到主窗口的原因。你希望打开主窗口时让那些宏运行。在主窗口中插入新宏就是靠向工程文件的** CONFIG **节中加入宏来实现的。向其它窗口中加入宏会改变那些窗口的显示，例如：如果向一个辅助性窗口中加入浏览功能，需要加入一个或多个宏以设置一定的条件，从而忽略在** Buttons **属性页的** Browse **按钮上所作的设置。这些辅**

助性窗口在帮助工程文件中要专门设置 CONFIG＜窗口名＞小节。

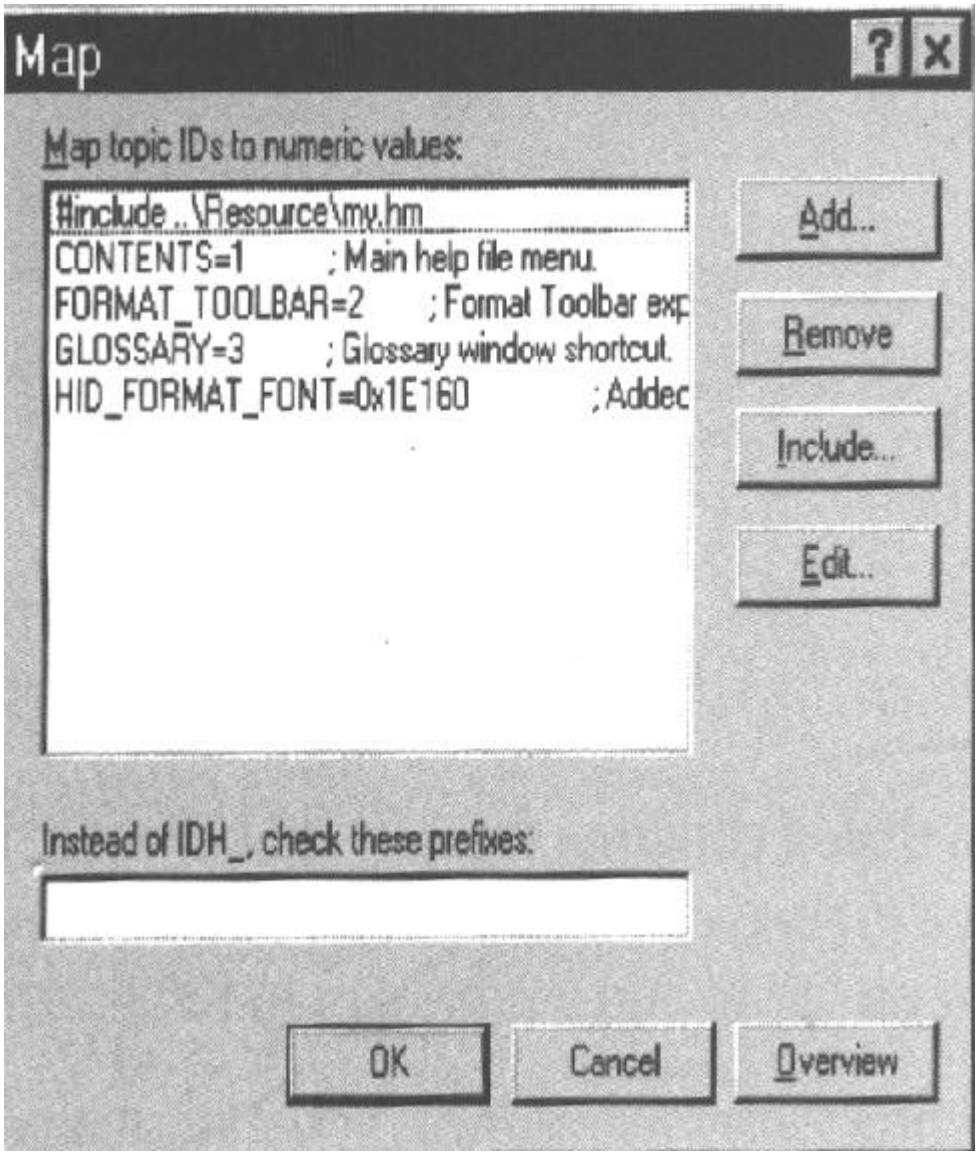
注 添加到 Windows Properties 对话框的 Macros 属性页中的宏通常只影响一个窗口，而不是帮助文件中的所有窗口。

技巧 访问主窗口 Macros 对话框的另一种方法是单击如图 15.2 所示的主窗口的 Config 按钮。

映射帮助主题

前面已经陈述过这部分的重要性，如果没有把帮助文件中的主题标识符映射到帮助内容序号上，就不能建立一个对应用程序中控件内容上下文相关的帮助文件。下面我们将讲述如何实现这种功能。

单击 Map 按钮，系统显示如下图所示的 Map 对话框。



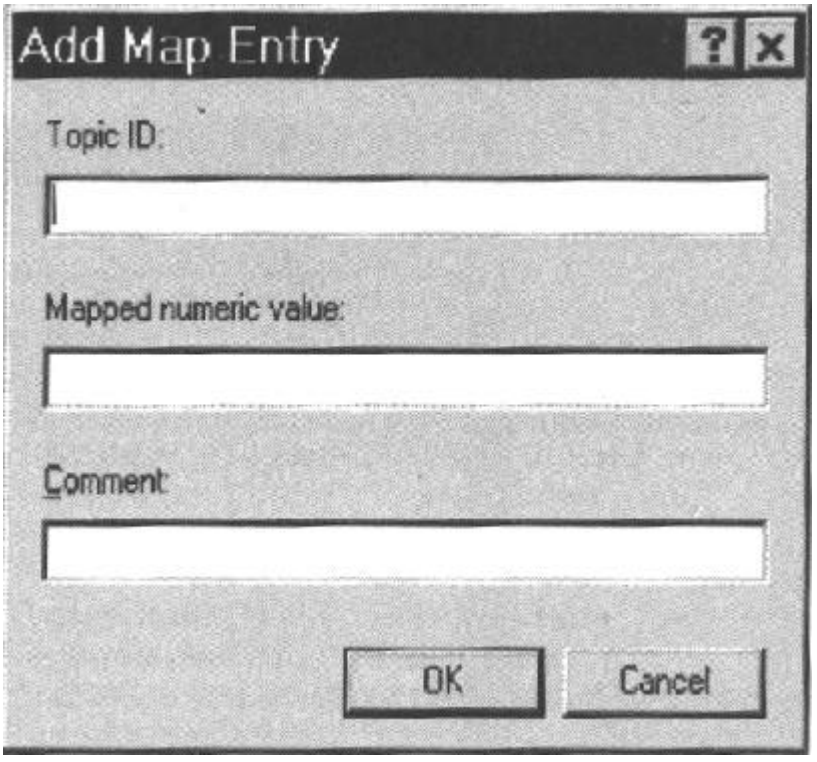
在这里可以定义主题标识符与内容序号间的关系。上图中，我已经定义了一些联系，主题标识符被设置为与帮助内容序号相同的值，紧跟其后的是描述这一项目的注释。

有许多方法可以直接保持内容序号，对于小的帮助文件我一般是从 1 开始编号，递增直到帮助中的最后一个主题。大帮助文件比较复杂，可能会碰到重

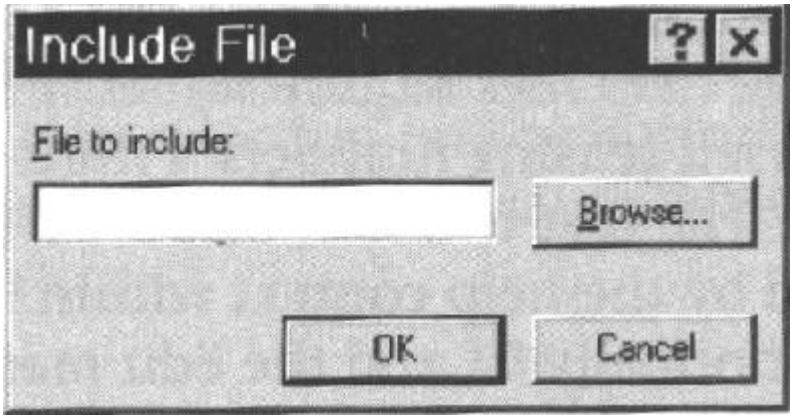
用序号的情况，这时，我一般用 3 或 4 位数的序号，前两位数是控件和菜单条在应用程序帮助内容中的位置。例如：菜单项 File 标为 01，菜单项 Edit 标为 02，File|New 命令的序号就是 0101，因为 New 命令项通常是 File 菜单中的第一项。第一个非应用程序主题赋值 0001，例如词汇表。应用程序窗体上控件的前两位数应大于最后一个菜单项的序号，我使用 Tab 序号作为控件标识的后二位数字，原因在于这类控件不像标签或其它的非活动组件，这些非活动组件不会出现在帮助文件中。

要增加新的映射入口页很容易，单击 Add，你会看到如下图所示的 Add Map Entry 对话框。

该对话框有三个域：主题标识符、被映射的数值（帮助内容中的序号）及注释。填完后单击 OK 就在工程文件中加入了一个新的映射。



正如前文所述，帮助文件中一般都包含 HM 文件，以减少工作量并提供快捷便利的检查方法。当帮助文件已测试并发行之后，才发现有一些很重要的主题没有覆盖到，再没有什么比这更难办的了。这时单击 Include 按钮，系统显示 Include File 对话框，如下图所示：



这个对话框提供了一个 Browse 按钮，可以用它调出已经保存在磁盘上的文件。单击后打开标准 Open 对话框，就像你在其它程序中用过的这种对话框一样操作。

编译帮助文件

完成工程文件后，就要进行编译了。如图 15.3 所示，单击主窗口底部的 Save and Compile 按钮。在编译过程中，Help Workshop 窗口会最小化，这样你就可以去做别的工作了（因为编译大的帮助文件需要很长时间）。编译完成后，会看到如图 15.4 所示的对话框：

你应该注意到这个对话框中立刻显示了一些东西——上图显示了帮助文件中存在一些错误（实际上那些都是注释，但绝大多数情况下还是应该把它们看

成是错误）。在帮助文件条目中，我们没有给出 HM 文件中的所有条目，即：在帮助文件覆盖过程中出现了漏洞。从中我们可以看到，使用 HM 文件很有帮助——至少会减少丢失帮助主题的机会。

那么 HM 文件就很完美了吗？远远没有。看一下表 15.2，那里曾经给 ID_FORMAT_FONT 增加了一个 Map 条目，这是 MFC 用在 AFXRES.H 文件中的标准标识符，不是包含定制控件标识符的 RESOURCE.H 文件，结果 HM 文件不能识别它——必须手工增加对标准控件的支持。

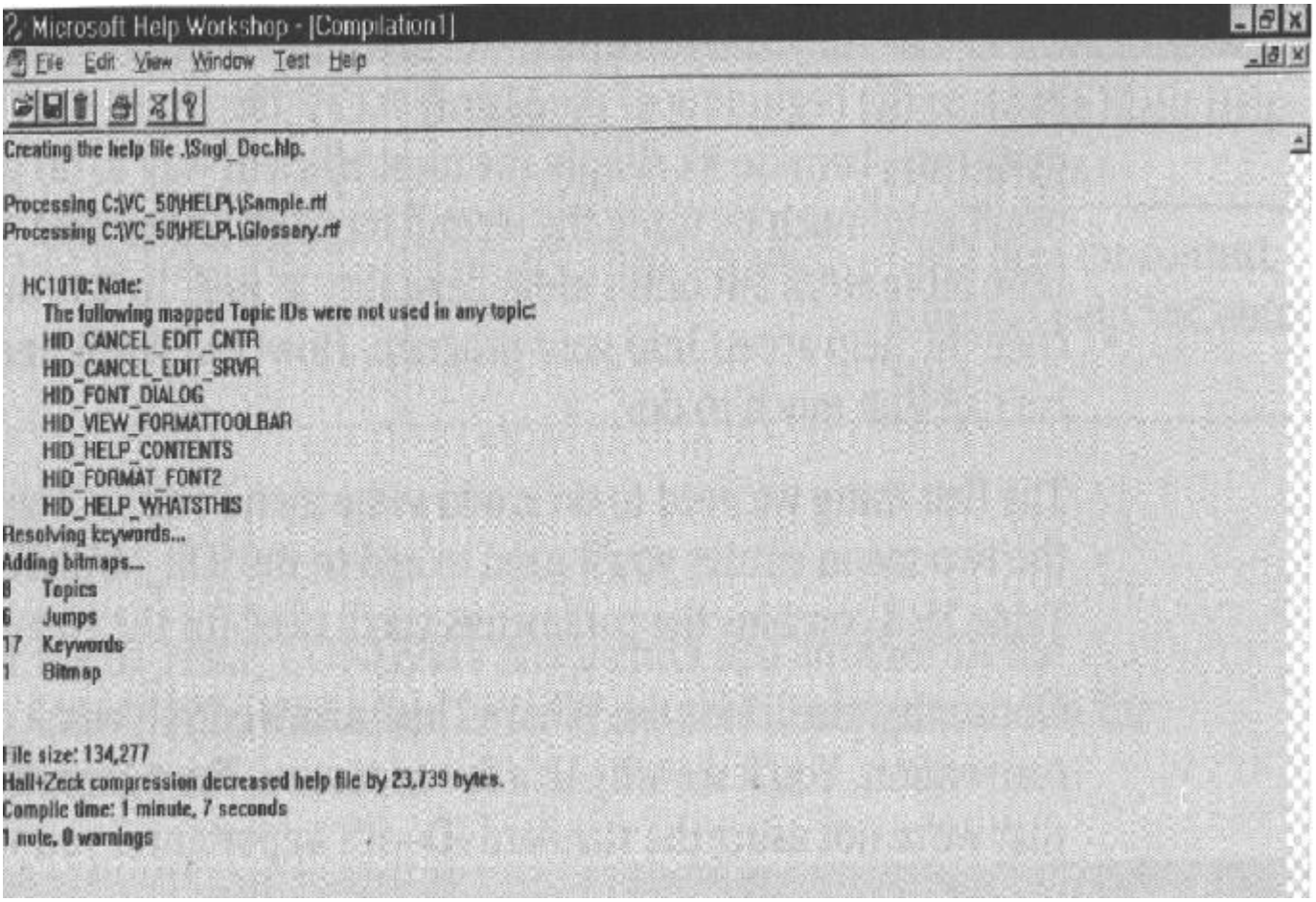


图 15.4　该编译结果对话框显示了帮助文件的当前状态及错误信息

15.5 为应用程序增加标准帮助

终于可以给我们的应用程序增加帮助了！这里选用第 2 章开始的单文档工程文件的资源版（第 3 章实现）作为例子，因为它非常完整，而且在以前我们没对它作任何帮助支持。必须认识到帮助支持的最重要部分在于帮助文件本身，我们要实现的帮助支持根本不需要多少编码。

实际上我们会看到两种实现方法，第一种方法是全自动的——只需在 MaiFrm.CPP 的开始处加入一个菜单项并修改适当的菜单项标识。这是最常用的方法，因为它最有效。第二种方法略微复杂些，但也更加灵活，需要在程序中手工加入访问帮助的代码，但实际上还是很简单的。

首先要增加一些菜单项。图 15.5 给出了要加入到 IDR_MAINFRAME 菜单中的两个菜单项，表 15.3 是要用到的参数。

表 15.3 帮助菜单参数

菜单标识（ID）	菜单标题	提示
ID_HELP_CONTENTS	&Contents\tF1	Display the Main Help File\nHelp
ID_CONTEXT_HELP	&What's This?\tShift-F1	Click here for context sensitive help. \nContext Help

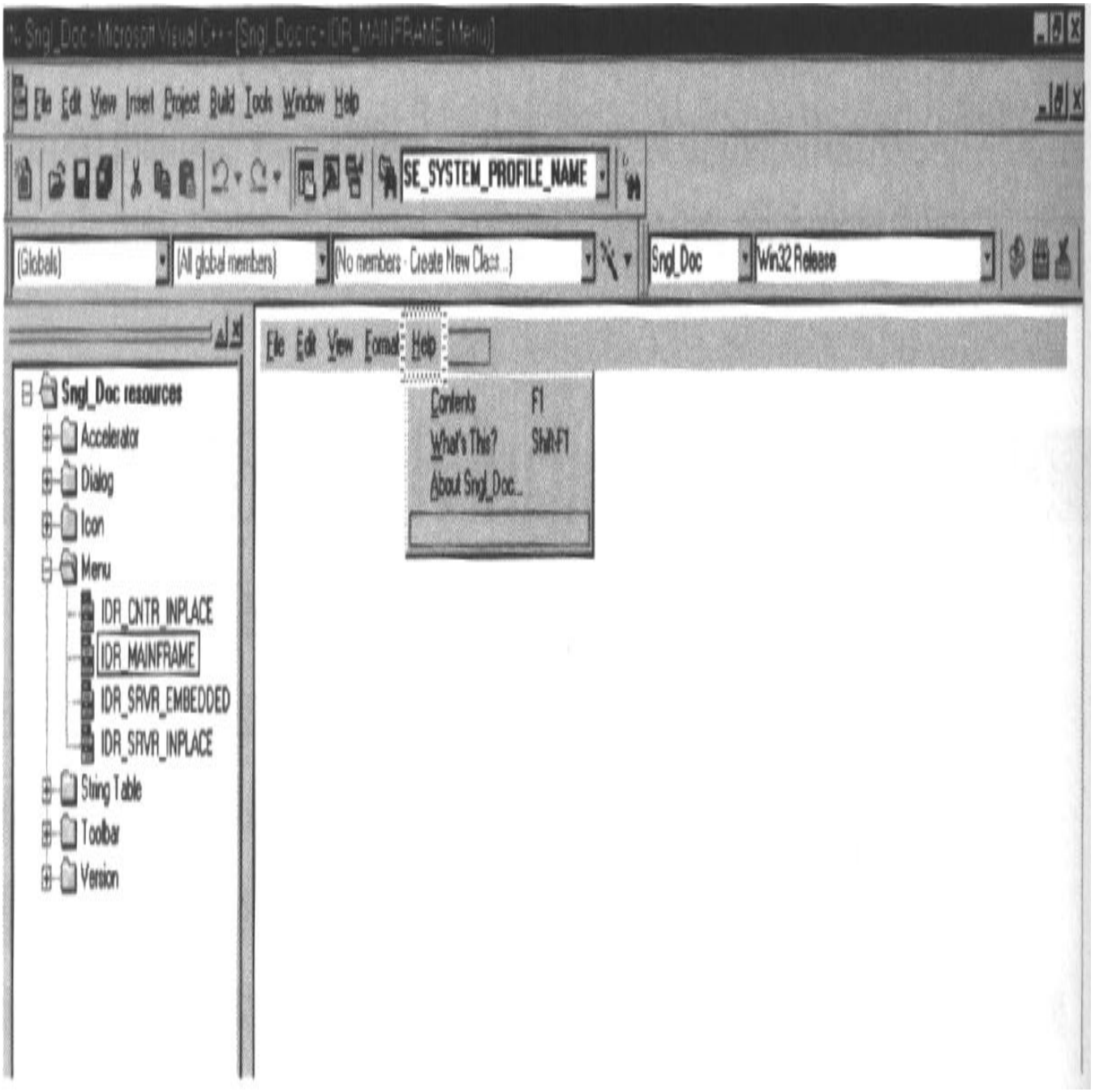


图 15.5　**需要增加两个新的菜单项来测试本章前一节创建的帮助文件**

注意 What's This?的 ID 号，这个菜单项没有使用通常的命名规则。一会儿就会讲到，现在，只需记住这一点——如果想自动实现 MFC 帮助，这个命名是非常重要的。

同时也要修改 IDR_MAINFRAME 工具栏上的 help 按钮，当前它显示的是 About 对话框，没有加入帮助支持时，这当然是很合适的；加入帮助时，打开 IDR_MAINFRAME 工具栏，双击 help 按钮，将 ID 号改为 ID_HELP_CONTENTS。

这两个菜单项当然也可以有与之相关的快捷键，所以还需要打开如图 15.6 所示的 IDR_MAINFRAME 快捷键定义窗口。这里我已经加亮了其中一个列表项，可以给 ID_HELP_CONTENTS 菜单项增加一个新的快捷键，给 ID_CONTEXT_HELP 菜单项增加一个快捷键，如下表所示。

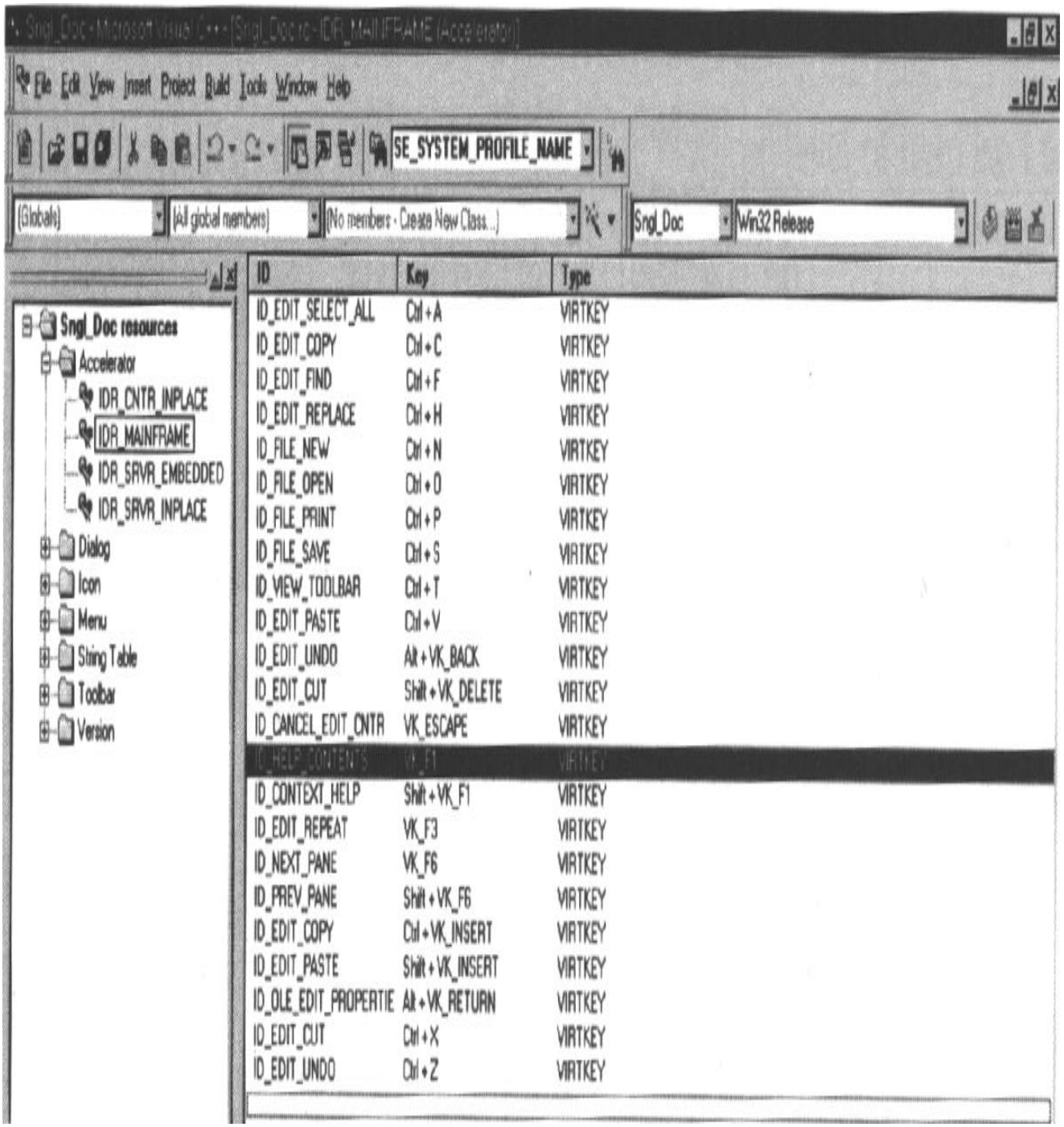
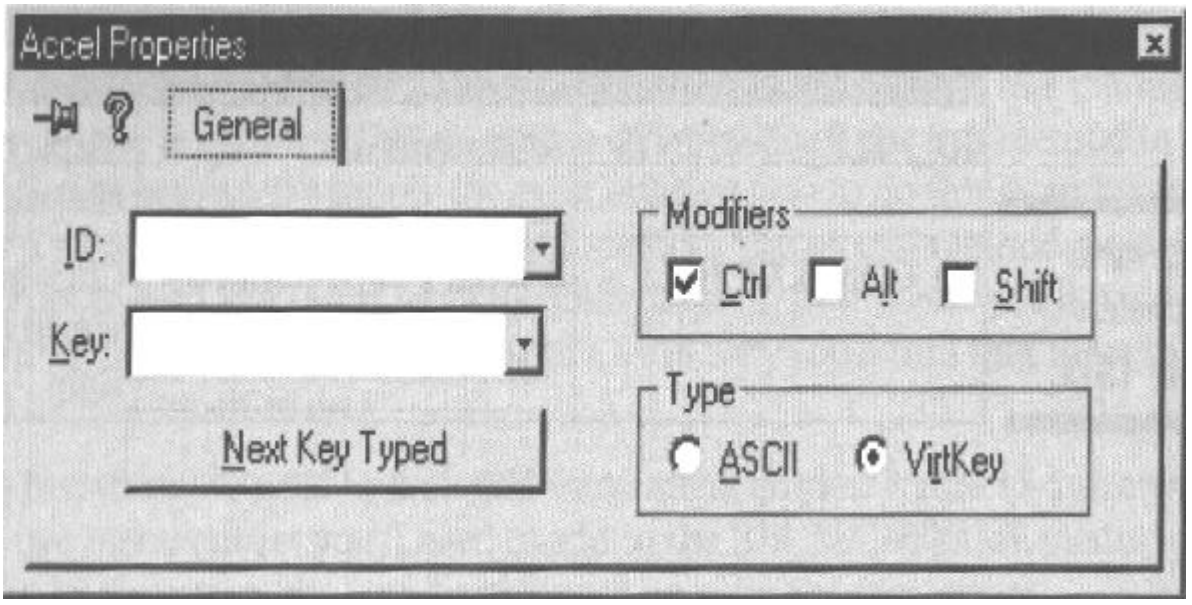


图 15.6　**需要为这两个帮助菜单项定义快捷键，以便使它们对用户更友好**

菜单标识（ID）	键	类型
D_HELP_CONTENTS	VK_F1	VIRTKEY
ID_CONTEXT_HELP	SHIFT+VK_F1	VIRTKEY

加入快捷键很容易，双击最后一个空白列表项处，显示如下图所示的 Accel Properties **对话框：**



从下拉 ID **列表中选中所需的** ID **号，单击** Next Key Typed **按钮，按下分配给该** ID **的热键，这里** ID_HELP_CONTENTS **的热键为** F1，ID_CONTEXT_HELP **的热键为** SHIFT-F1。**要确保选中了适宜的复选框，然后关闭该对话框。**

记住我们说过，不用写实际的函数就能实现这些帮助选项。MFC **支持四个根本不用编程的缺省帮助动作：内容（**Contents**）、查找（**Find**）、索引（**Index**）和上下文相关（**context-sensitive**）功能。我们用内嵌的** MFC **函数实现上下文相**

关功能。我们所要完成的全部工作就是加入程序列表 15.3 粗体部分所示的代码，该代码用于激活实现上下文相关功能的内嵌 MFC 函数。

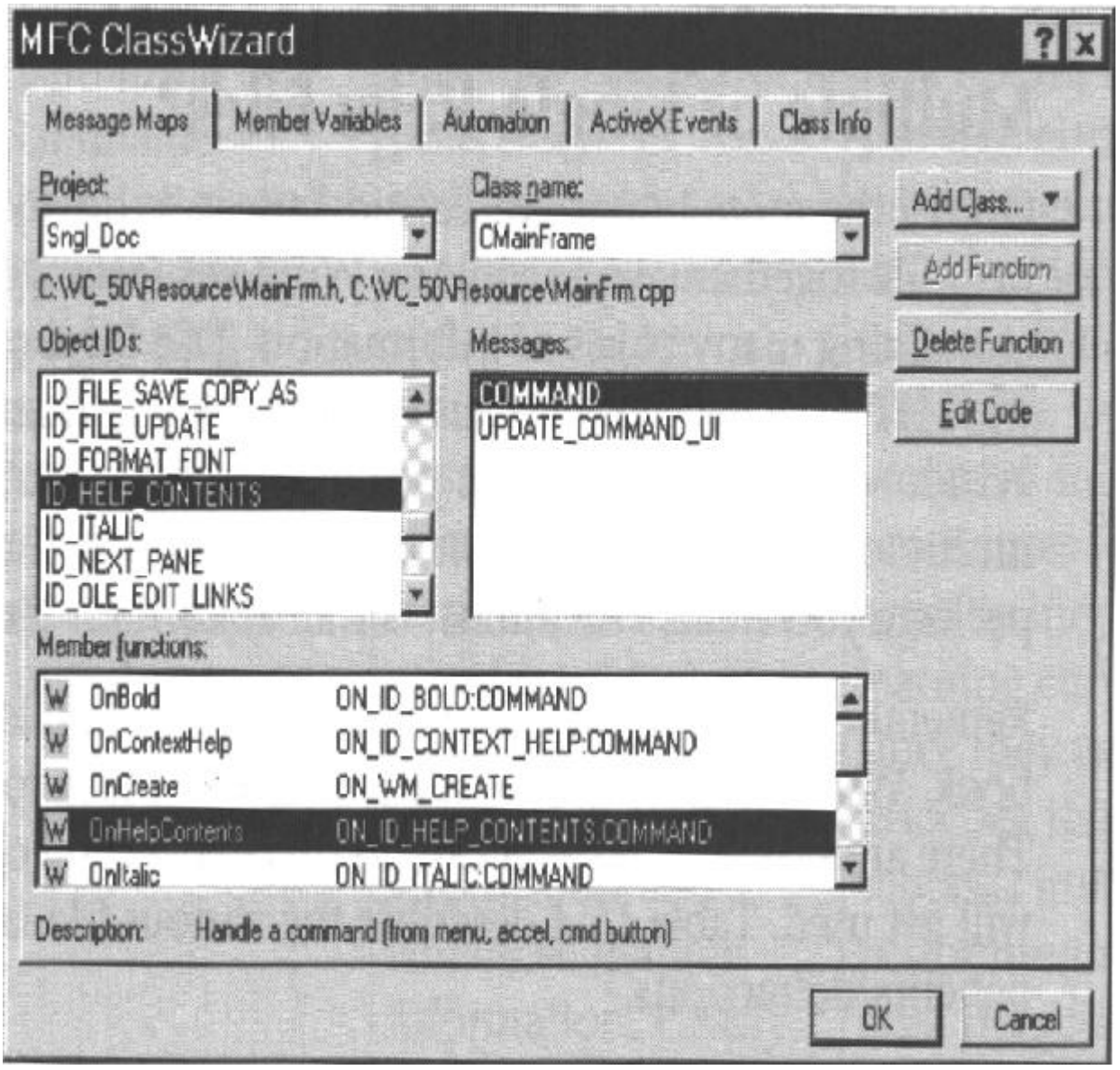
程序列表 15.3

```
BEGIN_MESSAGE_MAP(CMainFrame,CFrameWnd)
    //{{AFX_MSG_MAP(CMianFrame)
    ON_WM_CREATE()
    ON_COMMAND(ID_BOLD,OnBold)
    ON_COMMAND(ID_ITALIC,OnItalic)
    ON_COMMAND(ID_STRIKETHROUGH,OnStrikethrough)
    ON_COMMAND(ID_UNDERLINE,OnUnderline)
    ON_COMMAND(ID_VIEW_FORMATTOOLBAR,OnViewFormattoolbar)
    ON_COMMAND(ID_HELP_CONTENTS,OnHelpContents)
    ON_COMMAND(ID_CONTEXT_HELP,OnContextHelp)
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()
```

注　必须用选定的 ID 激活 MFC 帮助支持，如果在使帮助起作用时发生了问题，请检查菜单项的 ID 号。

现在要看一看 Contents 菜单项了。先要增加一个函数，用 View|ClassWizard 命令显示 MFC ClassWizard 对话框，选中 Message Maps tab 并选择 Class Name 域的 CmainFrame，在 Object IDs 程序列表中找到 ID_HELP_CONTEXT，在

Messages **程序列表中加亮** COMMAND **列表项，单击** Add Function**，此时的** MFC ClassWizard **对话框如下图所示。**



单击 Edit Code **按钮，就会看到函数框架，程序列表** 15.4 **显示了使** Contents **菜单项起作用所需的代码：**

程序列表 15.4

```
void CMainFrame::OnHelpContents()
{
    //Call WinHelp to display the main help window.
    WinHelp (0,HELP_CONTENTS);
}
```

正如你所看到的，实现所需帮助的编码并没有什么特别之处，说实话，这段代码可能是本书中最短的代码。你所要完成的所有工作就是以预定义量 HELP_CONTENTS 调用 WinHelp()。这里我们不需要提供任何辅助信息，因此第一个参数被设置为 0。

15.6 建立基于 HTML 的帮助文件集

第八章已给出了一些建立基于 HTML 帮助文件必需的知识，HTML 帮助的基础元素是 Web 页面，就像在 Internet 上一样。然而，HTML 帮助又不仅仅是一个简单的 Web 页面，还要考虑许多其它因素。例如，要从访问的角度组织帮助文件。以下各节帮助你设计一个既有效又实用的帮助文件。

采用多页还是单页文件

注 基于 HTML 的帮助文件经常用多个文件存放帮助信息；这些文件的分布

形式很大程度上影响了用户查找时的便利性。

一个很有必要考虑的因素是如何组织所有帮助文件。使用 Windows Help 时，我们只建立了一个文件，它由许多信息页组成。这种方法不适用于基于 HTML 的帮助文件，因为此时不是用主文件的方法来实现的，总而言之，必须指出如何将多个信息页放入一个集合中，但所建立的集合必须包含所有的文件。

请谨记，帮助文件使用大纲方式组织，而且编排上与书籍的布局格式相似。你需要完成的工作是确定是从哪儿截断文件，完成这项任务的方法有多种，关键取决于应用程序中如何使用，表 15.4 描述了各种文件布局选项以及相应的优缺点。

表 15.4　HTML 文件的布局方法

布局方法	描述	好处	缺点
大菜单/对话框控件	对话框中的所有控件或树状菜单中的全部菜单项都放在一个文件中，使用锚地的方法可以让用户从一个主题切换到另一个主题，这种方法包含一个目录，让用户	初次使用这种布局方式的用户会觉得很容易浏览整个程序，而且，这种设计使帮助文件与使用手册很相像，目录的使用方便了信息的查找	过长的文件可能会使用户觉得下载开销过大而转去下载其它文件，而老用户又会觉得在一个文件中多次选定菜单项很麻烦，而且，由于修改时可能要对多个文件做相同的改动，因此会造成更新困难

	直接切换到所需主题		
大任务	每个文件只含有对完成某一特定任务的完整描述和过程。例如：对建立新文档的操作建立一个帮助文件	初次使用这种布局方式的用户会觉得很容易完成具体任务，而且每项任务都有独立的完整性，减少了程序变动时带来错误的机率；而且可以包含屏幕快照和图像信息；最后，这是唯一一种可以产生简单的教程帮助文件的编排方式，由于 Internet 带宽和灵活性的限制，降低了我们能提供的培训的能力	高级用户会觉得，经过许多程序项目才能查找到一条所需的信息很麻烦。缺少常规的子程序会增加用户的学习难度，因为他们不能把执行某一特定任务和所要调用的常规子任务联系起来。由于有冗余信息，帮助文件难于更新，并占用更多的服务器空间；增长的文件长度也使远程下载更加费时
小菜单/对话框控件	每条菜单项或对话框中的一个控件放在一个单独的文件中	较短的下载时间使得用户即使使用拨号方式上网也会愿意使用这种帮助。由于用户只看到一个屏幕会觉得查找信息很快。由于信息只出现	由于每条菜单项或对话框中的每个控件相应的信息都需要下载，初次使用的用户的积极性会大受打击，特别是用拨号上网时。目录不再是自动的，

		在一个文件中，更新快捷可靠。帮助文件在服务器上的总量减少了	需要单独的维护时间
小任务	建立任务清单和内容的树型结构表。每个任务被分成一系列易于实现的常规子任务，在任务总述中包括全部的特定事件	使用目录使得信息便于查找，特别是对于高级用户。树型结构形式使得用户可以直接找到所需信息。由于减少了下载时间，用户会更加愿意使用这种帮助。更新由修改每个子任务文件夹下的信息完成	可能会在任务处理过程带入微妙的错误，例如：菜单项可能在应用中的某一部分已经改变了，而另一部分则没变。一些新用户则会觉得面向任务的方式难于理解
主题型	每个文件包含一个帮助主题。例如：关于数据录入方法部分可以组成一个文件	想查找关于某个主题的用户会觉得十分方便，因为所有信息下载一次就完成了	对于信息组织方式依赖于菜单的远程用户，必须在所有菜单项信息下载完成后才能使用。因为一个菜单的变化需要查阅几个文件。信息更新也比较困难，原因在于一个菜单变化时就需要更新多个文件

连接类型

注 即使是基于 HTML 的帮助文件在为用户提供信息时，也必须考虑到拨号连接存在的限制。

一个很有趣的现象是：使用基于 HTML 帮助文件的厂家总是觉得用户是用 T1 线路接入 Internet 的，实际情况是用户常常是用拨号方式下载文件。在第八章我们已经讨论了一些关于拨号连接的情形，最重要的是要限制每个文件的大小，这样用户才会愿意下载。

与浏览器不同，应用程序中不会含有 stop 按钮，这就意味着用户必须选择是等着每页从 Internet 服务器下载完再享用帮助，还是在帮助文件过长时拨打帮助热线。多数情况下我还是建议你组织好一个强大的技术支持小组，准备为那些用了长 HTML 帮助文件的应用程序提供支持。

注 在基于 HTML 的帮助文件中要慎用图像，因为这会极大地延长下载时间。

考虑 HTML 帮助文件的大小意味着：需要考虑是否使用响铃和报警来增强基于 HTML 的帮助文件的效果。例如：大部分 Windows 帮助文件使用全彩色的屏幕快照以帮助用户获得最易理解的帮助信息；另一方面，对于基于 HTML 的帮助文件就使用了黑白方式的图片以减少下载时间。

还需要控制 HTML 帮助文件中图像信息的使用数量。在颜色和/或分辨率上的减少毕竟是有限的，如果图像因此变得很不易识别，用户就很不值得下载了。换句话说，不论建立哪种形式的帮助都必须保证图像的质量。

结束这种无益争论的最好办法是先进行用户调研，一些用户可能只需要文字，可以在帮助屏幕上作一个选项供他们选用；另一些用户宁愿使用低质量的

黑白图片，还有一些用户则选择高质量的彩色图像，因此最好提供多种方式供用户选择。这时要考虑到用户的拨号速度，毕竟有些用户确实用 T1 连接方式，他们应该得到高质量的帮助文件。

查找能力

当用户经过了初级阶段之后，就很关心帮助文件提供的对所需信息的查找能力。他们要找的可能只是完成某项任务中的一小步，或是某个对话框中一个小控件的局部说明。糟糕的是，HTML 帮助文件不是全局可查找的，这个问题是所有供货商都必须面对的。

有些方法可以模拟 Windows Help 的查找方法，实现起来也不难。虽然不能完全实现 Windows Help 的查找能力，但可以尽量接近。下面是一些供参考的方法：

- 建立索引
- 定义附加的链接
- 增加目录
- 生成热门主题列表

注　基于 HTML 的帮助文件缺少按字查找的能力。

基于 HTML 的帮助文件不像 Windows help 那样，它缺少按字建索引的能力。这就意味着用户可能会提出许多预想不到的问题，而帮助文件的查找功能又无法问答。这时一般要提供某种形式的反馈机制，以便于用户提出其它链接的需求。另外，如果你的 HTML 帮助文件查找功能足够灵活的话，建立附加链接不

会是件频繁的事。

技巧 克服 HTML 帮助文件查找能力不足的一种方法是使用 Windows NT IIS version 4.0 及以上版本。它提供了对 Web 目录下所有文件建立索引的能力，在基于 HTML 的帮助文件上设置查找页面并不广受欢迎，特别是使用 Microsoft 提供的通用查找页面。

15.7 向应用程序增加基于 HTML 的帮助

向应用程序增加基于 HTML 帮助过程分几个步骤完成，前面几节已经讨论了第一步：考虑 HTML 文件的设计和布局。下一步是建立 HTML 文件本身，最后在应用程序中加进必要的链接。

在第八章中，我们已经讨论了建立 HTML 文件的基础知识，这里不再赘述。当然要想从脚本方便地转化为好的帮助文件，还不得不借助于一个适宜的编辑器，比如 FrontPage 或 Visual InterDev。这时，脚本用 Word for Windows 编写也无妨，否则需要附加的产品如 Map This 在图像和各种帮助主题间建立链接。这个过程相似于我们前面做过的用 Hotspot Editor 执行编辑的过程。

我们还用前面章节中用过的作 Window 帮助的例子为例，唯一的区别是用 Word for Windows 将它转换成了 HTML 格式。理论上说，可以使用任何工具建立帮助文件，但用 Word for Windows 建立的主文档既可以用于印刷出版物，又可以用于电子出版物。我们还用那节中同样的示例程序，但把原始的资源工程

文件拷贝到另一个目录以保持帮助例子的独立性（屏幕快照上的目录名可能与机器上实际的目录名不匹配）。

注释 可以在以下站点 http://www.osborne.com 找到本书中的所有工程文件、帮助文件的拷贝。它们会帮助你更好地准备帮助文件。

与本章中 Windows 帮助格式的示例不同，没有自动实现向应用程序中加入基于 HTML 的帮助文件的方法，只能为帮助文件中的每个访问点手工编写代码。我们的例子看上去很像本章第一节中的 Windows help 访问的第二种形式，而不是第二节讲述的方法。

这个示例由在帮助菜单中增加两个菜单项开始。精确的过程可以参见前面小节讲解的如何向应用程序加入标准帮助的方法来实现。菜单项的格式如图 15.4 所示，表 15.3 则给出了用到的参数。当定义了菜单项后，分别给 ID_HELP_CONTENTS 和 ID_CONTEXT_HELP 菜单项添加新的快捷键，如图 15.5 所示（前面在加入标准帮助时，我们已经讲述过了）。

注释 本章介绍了在应用程序中使用基于 HTML 的帮助文件的一种很传统的方法（不包括上下文相关帮助，但添加上下文相关帮助的方法与添加基于 HTML 的帮助的方法相似）。还有其它增加基于 HTML 帮助方法，在 12 章中我们会讲述，那时，你可以不用本例中所用的访问主页的方法而是在 Web 站点上设置一个帮助桌面。

首先，在应用程序中加入一个新的对话框。在 ResourceView 中右击 Dialog 文件夹，从上下文相关菜单中选择 Insert Dialog，这时会看到一个标准对话框，

右击该对话框，从上下文相关菜单中选择 Properties，系统显示 Dialog Properties 对话框，将 ID 域改为 IDD_HELP_DLG（或其它你喜欢的名称作为帮助对话框的名字），将 Caption 域改为 Main Help（或其它你喜欢的显示在对话框标题栏的文字）。本例中还需要改变对话框的某些其它属性。在 Styles 属性页选中 Minimize 和 Maximize 复选框，还需选中 Border 域的 Resizing 选项，使用户能够重新调整结果对话框为适宜的大小。关闭 Dialog Properties 对话框。

现在有了一个用于显示的新对话框，这时要加入一个 Web Browser 控件。使用 Project|Add to Project|Components and Controls 命令来显示 Components and Controls Gallery 对话框。双击 Registered ActiveX Controls 文件夹，加亮 Microsoft Web Browser 列表项，单击 Insert 向工程文件中加入这个控件，当提示是否确定时选择 OK 按钮。完成上述工作后，关闭该对话框。

本例要用到 5 个命令按钮和一个 Web Browser 控件。有时一图达千言，所以用图 15.7 来说明我对帮助对话框所做的配置。这是使用基于 HTML 帮助时你应该向用户提供的最小必要配置，原因在于用户总是需要在各个帮助屏幕来回翻页的。表 15.5 给出了该对话框中所有控件的属性列表。

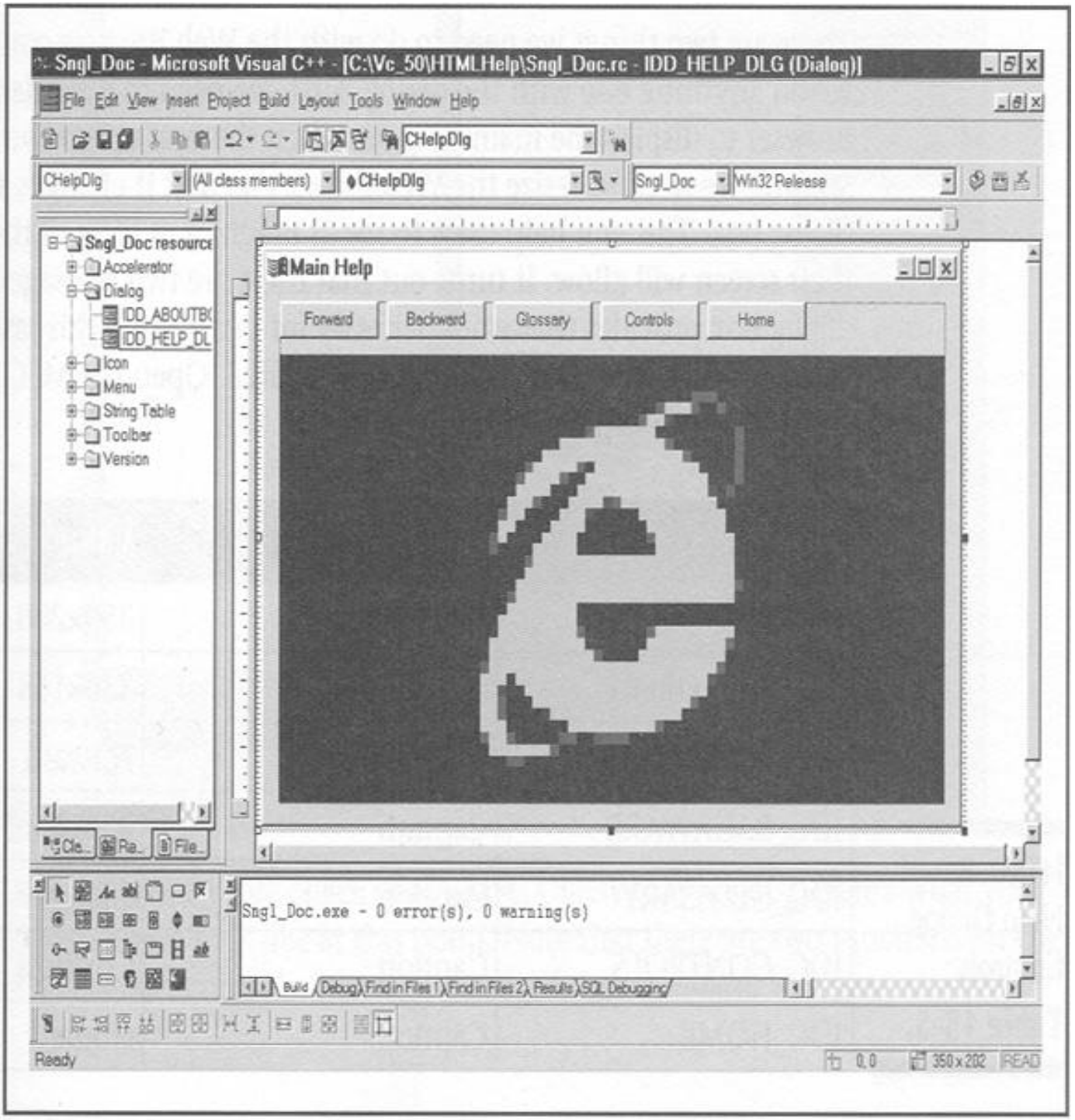


图 15.7　**一定要把帮助对话框设计得易于使用**

表 15.5　基于 HTML 的帮助对话框控件

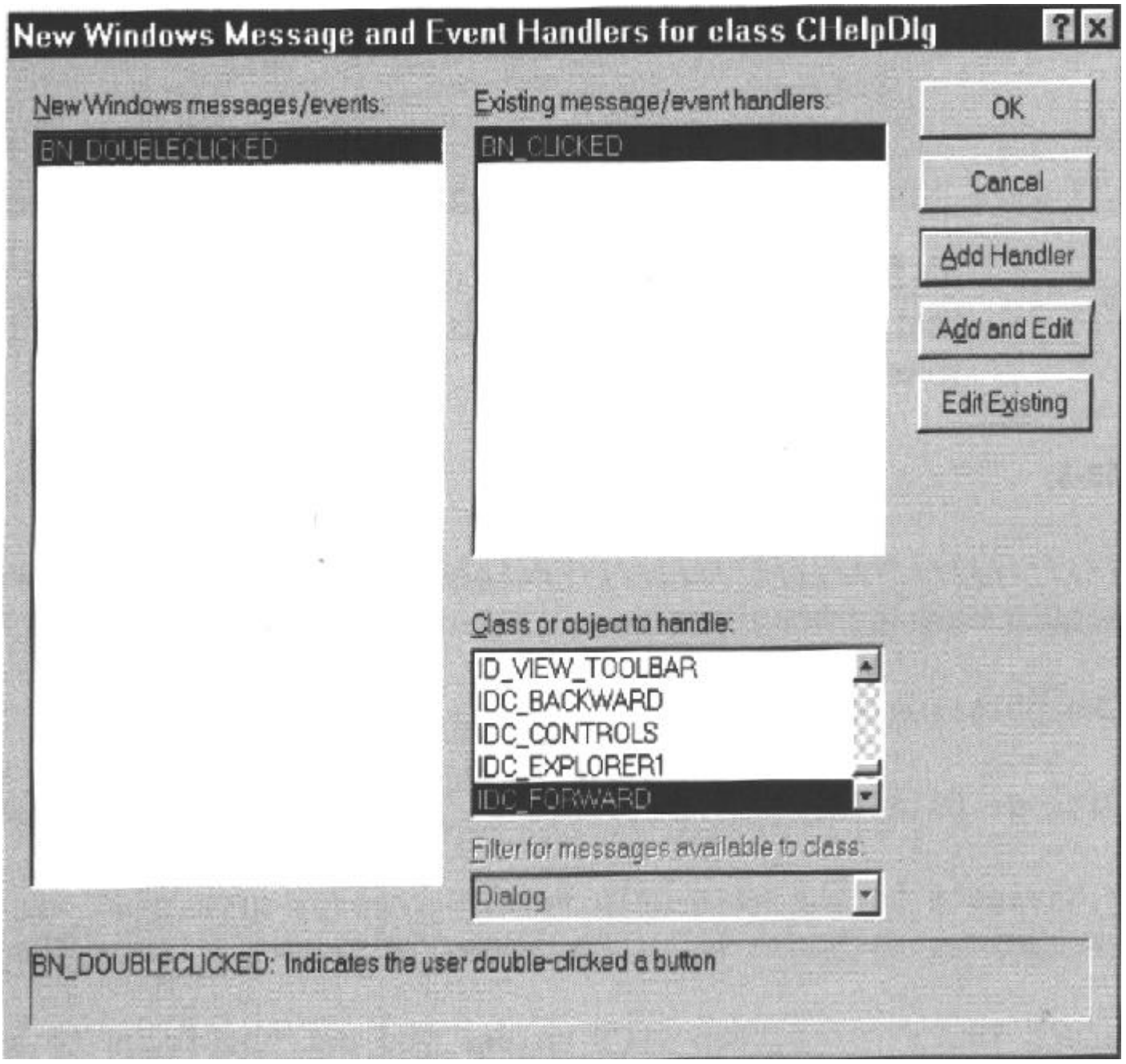
控件	属性	取值
IDD_HELP_DLG	Size	350x200
IDC_EXPLORER1	Size	336x168
IDC_FORWARD	Caption	Forward
IDC_BACKWARD	Caption	Backward
IDC_GLOSSARY	Caption	Glossary
IDC_CONTROLS	Caption	Controls
IDC_HOME	Caption	Home

建立一个对话框并不代表在应用程序的各个部分都可以访问它，需要为它建立一个类，这也是我们将要完成的工作。按住 CTRL 键，然后双击该对话框（同时保证没有双击其中的任何控件），系统显示 Adding a Class 对话框，单击 OK 建立新类，输入 CHelpDlg 作为类名，并选择 CDialog 作为基类，按 OK 创建新类。关闭 MFC ClassWizard 对话框（在完成下面的几个安装步骤后再加入所需的代码）。

此时，虽然该对话框已经可以访问了，但对话框上的控件还都不能访问。我们需要处理的第一个控件是 Web Browser。双击 IDC_EXPLORER1 控件，系统显示 Add Member Variable 对话框。在 Member Variable Name 域中键入 m_WebBrowser，在 Category 域中选择 Control，在 Variable Type 域中选择 CwebBrowser2，单击 OK 完成变量创建工作。做上述工作的目的是为了在应用程序中使用 Web Browser 控件。

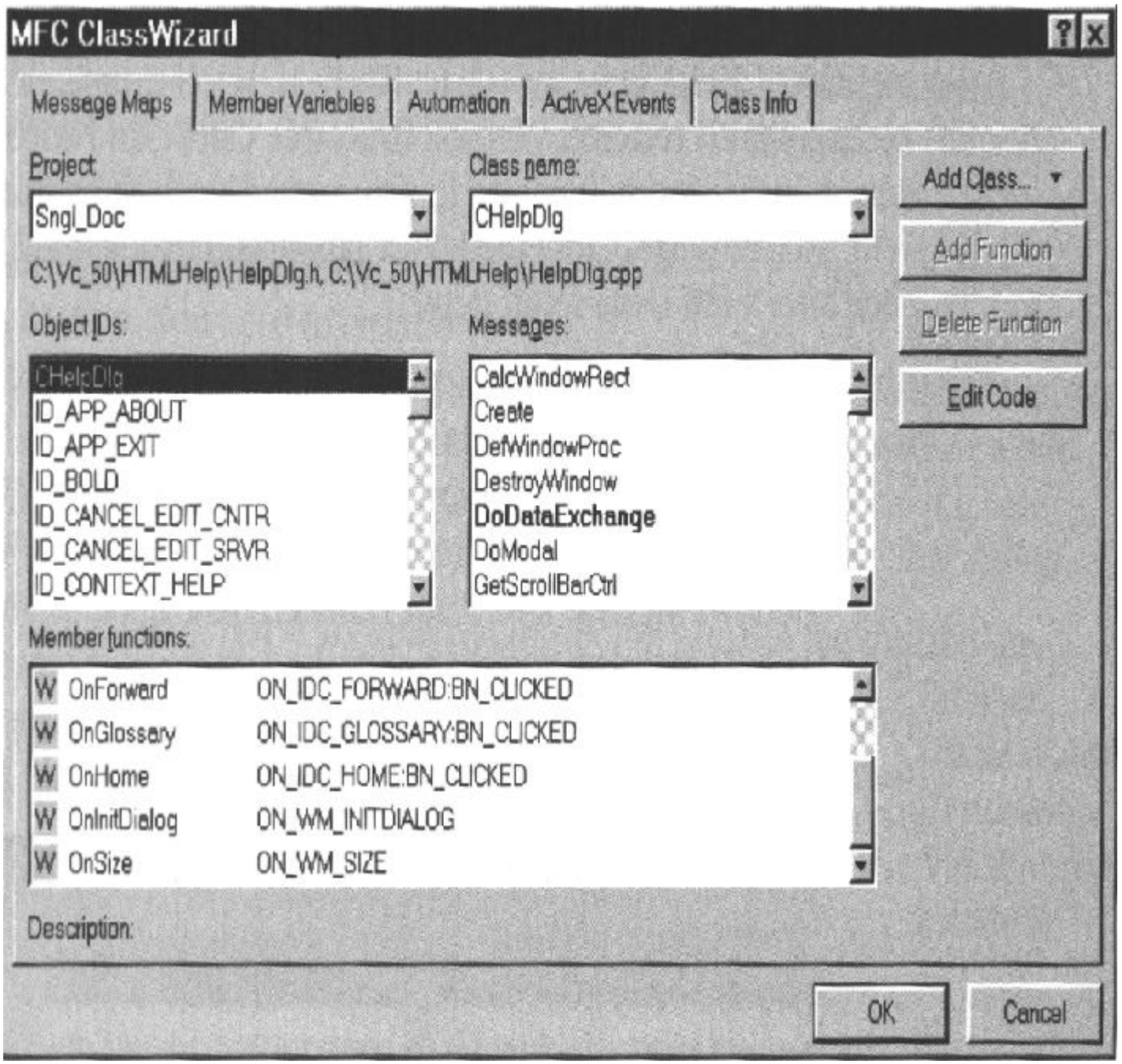
在处理该对话框的其它任务前，对 Web Browser 控件我们还要完成两项任务。其一是，我们需要初始化 Web Browser 控件，以便在用户打开该对话框时它显示主帮助文件。其二是，我们需要改变 Web Browser 控件的大小，以便它与对话框的大小一致。这样用户就能够根据屏幕大小和帮助文件的大小看到尽可能多的帮助内容。完成这项任务只需要使用对话框提供的两个事件，一个是 WM_INIDIALOG，另一个是 WM_SIZE。使用 View ClassWizard 命令打开 MFC ClassWizard，选择 Message Map 选项卡，然后加亮 Object IDs 域中的 CHelpDlg，滚动 Message 列表，直到找到 WM_INIDIALOG，单击 Add Function 在你的代码中添加缺省的初始化函数。对 WM_SIZE 消息完成与 WM_INIDIALOG 消息相同的工作。关闭 MFC ClassWizard 对话框（在完成了随后的几个设置步骤后，我们将添加所需的代码）。

用户当然需要五个按钮执行一定的任务，实现起来相对比较简单：右击第一个按钮（也就是 Forward 按钮），从上下文相关菜单中选择 Events，你会看到 New Windows Message and Event Handlers 对话框，加亮 BN_CLICKED 消息，然后单击 Add Handler 按钮。关闭 New Windows Message and Event Handlers 对话框。对所有五个按钮重复上述操作步骤，下图是 New Windows Message and Event Handlers 对话框的外观：



让我们再看一看 MFC ClassWizard **对话框，下图是当前状态下该对话框的显示信息。请注意，**Web Browser **控件有两个函数，每个命令按钮都有一个函**

数。



现在到了给我们的示例工程中添加代码的时候了。程序列表 15.5 显示了示

例程序中需要添加代码，在你看过之后我会再进行讲解。

程序列表 15.5

```
////////////////////////////////////////////////////////////////////////////////////////////
//CHelpDlg message handlers
BOOL    CHelpDlg ::OnInitDialog()
{
        CDialog :: OnInitDialog();
        //Navigate to the main help screen.
        m_WebBrowser.Navigate("http://nt_server/help2/sample.htm",              NULL,NULL,
NULL,NULL);
        return TRUE;        // return TRUE unless you set the focus to a control
                            // EXCEPTION:OCX Property Pages should return FALSE
}

void    CHelpDlg ::OnSize(UINT    nType,    int    cx,    int    cy    )
{
        CDialog :: OnSize( nType,    cx,    cy    );
        // Change the WebBrowser control size to match the dialog.
        m_WebBrowser.SetHeight( cy - 64 );
        m_WebBrowser.SetWidth( cx - 28 );
}
```

```
void   CHelpDlg ::OnForward()
{
      // Go to the next site in the history list.
      m_WebBrowser.GoForward();
}

void   CHelpDlg ::OnBackward()
{
      // Go to the previous site in the history list.
      m_WebBrowser.GoBack();
}

void   CHelpDlg ::OnGlossary()
{
      // Go to the Glossary Web page.
      m_WebBrowser.Navigate("http://nt_server/help2/glossary.htm", NULL,
                                    NULL, NULL,NULL);
}

void   CHelpDlg ::OnControls()
```

```
{
    // Go to the Controls (Format) Web page.
    m_WebBrowser.Navigate("http://nt_server/help2/format.htm", NULL,
                                    NULL, NULL,NULL);
}

void  CHelpDlg ::OnHome()
{
    //  Navigate to the main help screen.
    m_WebBrowser.Navigate("http://nt_server/help2/sample.htm", NULL,
                                    NULL, NULL,NULL);
}
```

正如你所看到的，这些代码丝毫没有神秘之处，特别是当你阅读过 12 **章的例子之后更是这样。所有的函数都与所建立的** Web Browser **内存变量（**m_WebBrowser**）打交道。在** OnInitDialog()**函数和** OnHome()**函数中都使用了** Navigate()**函数，使浏览器显示主帮助文件（**SAMPLE.HTM**）类似地，**OnClossary()**函数和** OnControls()**函数使用** Navigate()**函数来显示指定的页面；**OnForward()**函数和** OnBack()**函数则顾名思义，它们调用了** GoForward()**和** GoBack()**使得用户在** Web **页面间切换。最后一个函数是** OnSize()**，它用** Web Browser **控件的** SetHeight **和** SetWidth **属性来匹配对话框的大小。当用户改变对话框大小时，**Web Browser **控件的大小也随之而变。**

这样，我们就有了一个功能完善的帮助对话框了，但还是无法访问它。将它加入应用程序是很简单的，首先，像下面的黑体部分那样在 MainFrm.CPP **文件中加入帮助对话框：**

```
//    MainFrm.cpp    :    implementation of the CMainFrame class
//
#include    "stdafx.h"
#include    "Sngl_Doc.h"

#include    "MainFrm.h"

// Include our help dialog.
#include  "HelpDlg.h"

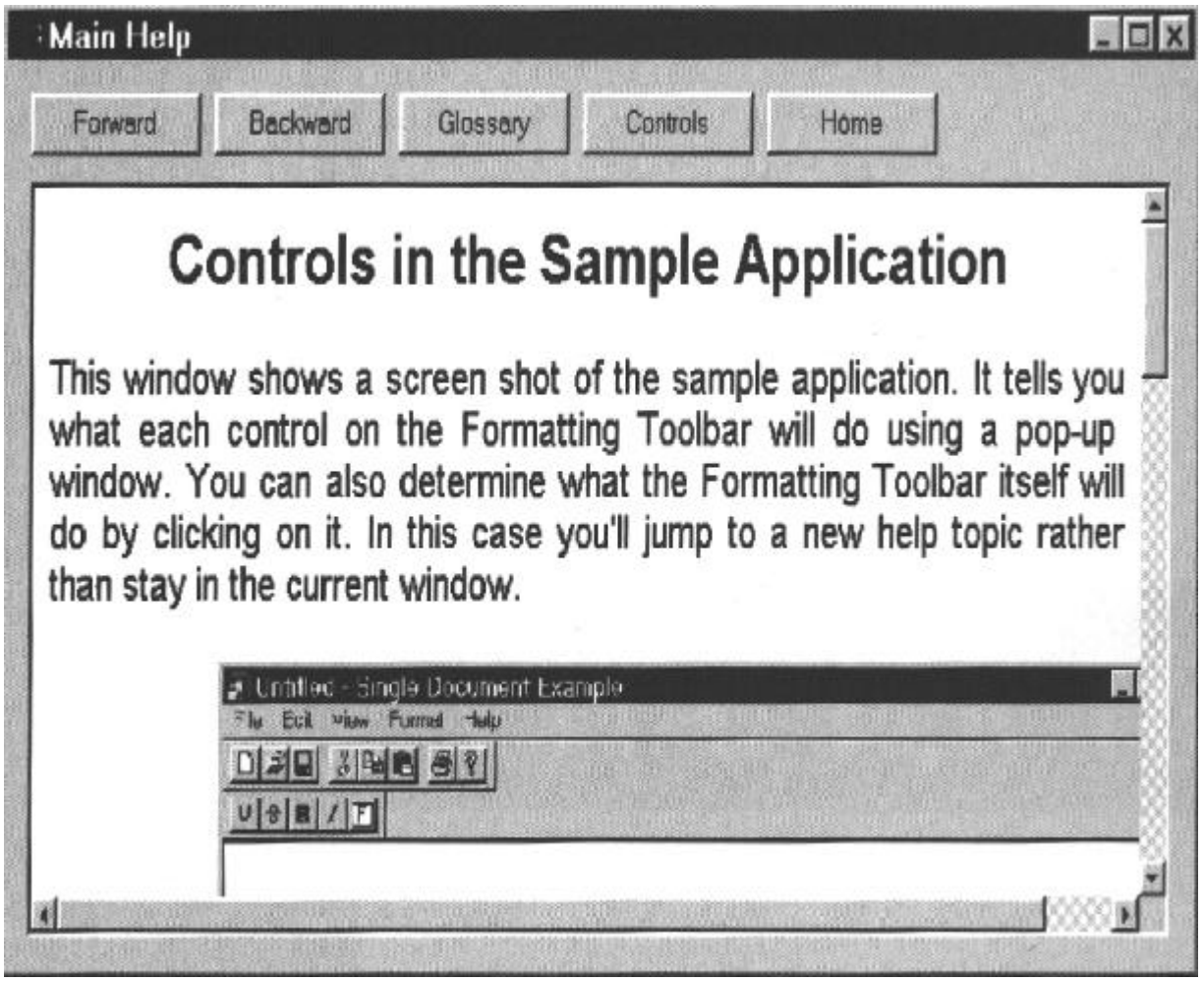
#ifdef    _DEBUG
#define    new    DEBUG_NEW
#undef    THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif
```

我们还需要在 OnHelpContents()**函数中添加一些代码，具体代码为：**

```
void CMainFram :: OnHelpContents()
```

```
{
    CHelpDlg    OHelpDlg;  //Create a help dialog instance.
    //   Display our help dialog.
    oHelpDlg.DoModal();
}
```

如果此时编译并运行这个应用程序，你就可以使用 Help|Contents **命令访问基于** HTML **的帮助。下图是帮助对话框的一个示例：**



第 16 章　打包应用程序

“打包应用程序”，听起来就像是打包物品送到商店中一样。从程序员的角度来看，打包与商店无关，但对用户使用时的感觉却至关重要。无需讳言，商店中最吸引我们注意力的是商品的包装，只有当打开包装之后，用户才会关心其中的内容。如果用户在安装应用程序时，就发现想要的东西都具备了，就会很愿意接下去继续使用——好的开始是成功的一半。

打包对于用户能够快速地安装程序、轻松地使用程序也很重要。设想一下一辆没有仪表盘的小矮车，它肯定能够前后左右地调整方向，但没有仪表板指示灯告诉司机正在走哪个方向；理由是设计师当然知道运动的方向，还安装仪表板灯干嘛!？这样的汽车肯定卖不出去！同样道理，一个没有好的帮助的应用程序，看上去像是让用户猜测能用它来干些什么——因为如果只是简单地加入了一个 README 之类的文件，用户是根本不会去看的。

注释　第 15 章讲述了建立良好的帮助文件的必要性。如果你还没有给你的应用程序作一个好的帮助文件，那么在阅读本章之前你应该首先创建一个帮助文件。我是好的帮助文件的积极拥护者——它们会节约你自己和用户大量的时间和精力，本章的前提是假定你也是好的帮助文件的拥护者。

注　要保证应用程序界面的实用性和一致性（包括安装程序）。

有些程序员只是简单地将所有程序文件放在一个软盘上，然后作一个批处理文件，就万事大吉、开始发行了；更有甚者，如果该程序只是内部应用，甚至连测试都不测试。过不了多久，他肯定会接到许多不满的用户电话，软盘上可能少了某个重要的文件，用户执行批处理后根本运行不了。没有人会再喜欢这种程序——即使是重新加入了一个好看的界面、安装程序并修正了错误。其实并不是程序本身有什么问题，而是程序设计者的态度不够认真。第一印象对一个程序的成功是很关键的！

不管谁用你的程序，都要在程序出门之前完成正确的包装。少做一点都会成为困扰你及用户的隐患。没有人愿意买一个不完整的小轿车——同样的道理，也没有人想要一个不完整的程序。花点时间做一个好的打包吧！——即使程序是给其它的程序员或高级用户使用的。人人都会受益于一个功能完善的程序！

如果我没有花点时间讲述程序的打包就是我的失职，这就是本章的目的。在我看来，导致一个应用程序给用户留下深刻印象的最重要的两个因素是：理解各种打包类型，为相应的打包建立安装程序。我还会讲到不同环境下程序员应特殊考虑的因素，例如，内部程序极少考虑开发共享软件的人所要考虑的共享问题。

Web 链接　本章我们使用了 InstallShield 6（Visual C++ 6.0 提供的 Free Edition）。可以在 http://www. Installshield.com/ 站点找到 InstallShield 6 和许多有用的用于打包的产品。不幸的是，至少在本书写作时，该软件还不能免费下载，然而，可拨打(800)374-4353

或 (847)240-9111 以了解该 Free Edition 并获得其拷贝。

16.1 理解各种打包类型

在打包交付应用程序时，需要考虑许多事情。对绝大多数程序员来说最重要的是运行环境。内部使用的程序可能与通用程序考虑的侧重点不同，它可能没有时间完成一个完整的打包工作——对内部应用来说时间往往是至关重要的因素。

注　有三种基本的程序打包类型：企业内部应用软件、共享软件和商业软件。

本节将讲述三种最基本的程序打包类型：企业内部应用软件、共享软件和商业软件。你很可能只会去阅读其中之一，但最好还是也读一下其它类型的叙述，因为其中包含有许多有用的技巧和提示。而某些人更不应该满足于只了解某一种类型，例如：你是公司的顾问，你也就是半个内部程序设计人，因为所建立的大部分应用程序都是供公司内部使用的；同时，为了满足外部用户的需求，至少要提供一个共享软件级的打包封装（一些客户甚至想要一个商业软件级的打包，但这种情况相对少一些）。

提供哪种打包方式很难一言以蔽之，但也不是毫无依据，关键在于对应用程序的使用人群作出正确的评估。如果用户需要的只是运行稳定、不需要多少的修饰，使用企业内部应用打包格式就很合适了。

企业内部应用软件

设计企业内部应用程序的程序员最不用关心封装的华丽性，最需要关心程序的可定制性。事实上，企业内部程序不用靠精彩的画面、美妙的音响来吸引用户，界面只要功能完善、用户友好就足够了。如果你手边还有许多项目有待完成，即使费尽心机地设计了一个精美的安装程序，也不会赢得老板的赞赏。

另一方面，企业内部程序却常要考虑提供可定制性。对于可定制的程序，特别是数据库应用程序，特别之处并不只是要使用特定的文件，还需要定制注册项及进行许多其它的设置。建立一个在企业内部各种工作站上都运行良好、不需要设计人员帮助的安装程序是最重要的要求，必须花些时间手工调整用诸如 InstallShield 一类产品建立的标准安装程序。

企业内部安装程序不同于其它两类安装程序的另一个重要之处是发布时所使用的介质。共享软件和商业软件，通常使用软盘存放（现在越来越少了）。除非你所在的企业太落伍了，否则肯定会有 LAN 环境供发布程序，可以充分利用这个环境。同时使用最简单而有效的存储介质：CD 盘。你会发现如果将应用程序打包在 CD 上，用 LAN 来发布，建立和测试安装程序都会快得多。

技巧 InstallShield 6 Professional Edition 提供了通过 Intranet 安装应用程序的能力。不管公司的规模有多少个楼宇，都可以应用我们介绍的原理，通过 LAN 发布应用程序。在决定以软盘或 CD 的形式存储应用程序并手工发布之前，必须花时间研究 InstallShield 或其它封装程序提供的各种发布方法。对于企业内部应用程序来说，用 LAN 或 intranet 来发布最为有效。

在 LAN 上打包好应用程序之后，只需在登录服务器的主批处理文件中加入安装程序命令就行了。下次当用户登录时，批处理检查该应用是否已经安装；没安装，自动调用已放在 LAN 上的安装程序。

技巧 安装公司内部软件时，用 UNC 路径代替标准驱动器符的方法非常方便。源目录用 UNC 路径保证每个用户都能方便地访问所需的源文件。

在企业应用环境中，发布程序时还需要考虑一些因素，下面各要点就是为保证安装程序的可定制性所要考虑的。使用哪种方式取决于具体环境，如企业机构的设置以及用户的连网情况。

通用文件可以集中存放 应用程序的标准发布方式为将所有文件都安装到用户的机器上，这是共享软件和商业软件发布的唯一方法；而对于企业内部程序来说，可以把通用文件——比如 DLL——存放到文件服务器上。这种方法有两个好处：首先减少了对用户机器存储空间的要求；其次当用户在 DLL 中发现错误时，更改起来更快——只需改动服务器上的 DLL 就行了。这种方法也有两个缺点：第一，用户必须连网；第二，如果经常加载/卸载 DLL 必然会极大地增加网络流量。

绝对的优势 共享软件和商业软件的设计者无法预先假定应用环境的任何因素——源和目的都不确定：使用何种机器是个未知数，有时连操作系统也是未知的；而对企业内部应用软件来说，设计者则没有这种麻烦，所要顾虑的是网上机型互不相同。如果对所有机器都用同样的安装方式，可以指定一个缺省的目的目录；使用 UNC 路径名时，可以指定一个绝对的源目录，因为服务器路径名不会因安装次数的不同而变化。当

网络规模不大、工作站配置一致时，甚至可以规范到每台站点。总之，不需要给用户提供太多的安装选项，从而从根本上减低安装程序的复杂性。程序员所要完成的工作就是事先胸中有数。

不用设立配置选择项　所有的商业软件都提供三种安装时配置选项：定制安装（Custom）、典型安装（Typical）和最小安装（Compact）。定制安装允许用户选择程序部件；典型安装专为桌面用户设计；最小安装则是为膝上电脑用户设计的。多数情况下可减化为两项：膝上安装和桌面安装。其实，用这些词汇只会使用户费解最小安装的含义。

共享软件

共享软件的设计者在打包程序时所遇到的挑战最大。看看其中的一个主要问题：安装的大小。与拥有高速网络环境的企业内部软件不同，也不像商业软件那样可以用 CD 进行发布，共享软件的用户一般用低速 Modem 连接到 BBS、联机服务或 Internet 上。有足够的证据表明，当程序大小超过 1MB 时下载它的人数会急剧下降。

注释　本节所限定的文件大小只是指导性的，潜在用户愿意在你的程序上投入的盘空间和时间取决于许多因素，诸如人为的估计和程序已显示的价值等等。例如：有一个我很喜欢的图像打包共享软件，它要占 7MB 硬盘空间，压缩后也超过 1MB，但人们还是很愿意下载并使用它，因为它物有所值！众口皆碑就是人们愿意下载的最好动力。你需要考虑

的是人们是否愿意为你的程序掏腰包——如果你真想参与到共享软件市场的话，这是必须学习的经验。

好了，这些决定了程序空间的限制。那么如何克服呢？关键在于如何将产品市场化。多数成功的共享软件一遍又一遍地使用同样的图像和声音；换句话说，不会出现安装程序用了一种奇异的图像，而应用程序本身又用了另一种图像，共享软件的要旨是安装和使用的界面的一致性。事实上，使用精确的程序设计技术能使你有多次机会轻松地摆脱应用程序的图标（我们在第三章讨论过）。

像企业内部软件那样没有动人的图像或声音，共享软件也能赢得用户吗？情况不太一样。如果真是想吸引用户进一步购买，就必须在程序中加入一些靓点。没有人愿意付钱给一个看上去邋里邋遢的程序，即使它的功能很强大。显然，想让程序既短小又完美是件很难的事。

开发共享软件的另一个问题在资金方面。许多共享软件的作者只有一两个人，在他们等待产品售出的过程中不得不从事其它咨询或冒险性的工作来维持生计，他们没有充裕的时间，也不会像出品商业软件的大公司那样，有专业的美术或音响指导。因此，共享软件只要设计良好、占用空间合适（比如说 5MB 左右）并稍加雕琢就很令人满意了。

我们尚未讨论共享软件设计者必须面对的最大的问题，安装程序只是第一步。企业内部软件对应用环境有最全面的控制，可以控制到使用该程序的每台机器，而且设计者即使在机型检查和配置选项方面作的工作很少也会得到认可。商业软件的设计者情况相近，至少他们可以在软件包装盒上印上说明，规定运

行该程序的硬件要求，事实上，只要用户买了相应的商业软件就会意识到对硬件环境的需求。共享软件的设计者就没有这种保证了，它的用户的机型可是包罗万象的——甚至是一台老掉牙的 8088 PC。

这种无法预测会带来什么后果呢？首先，在程序中必须额外设计一段检测例程，以保证满足最小硬件需求。用户总是很少去读 README 文件（通常在这里指出最小环境需求），甚至根本不注意它，当他们安装失败时，却只会抱怨设计者，根本不会承认是自己的失误。因此，必须有检测程序。还需要提供灵活的、可供选择的配置选项，例如：当机器档次低时，用户可以只选用文本格式，摒弃所有的图像和声音。

注 企业内部的程序员能够全面地控制应用程序的运行环境，而共享软件的开发者则几乎不能掌握其程序的运行环境。

技巧 如果时间和财力充裕，设计一个卓越的安装程序，非常有助于共享软件的销售。要让用户明白钱花得物有所值，换句话说，如果在安装过程就吸引了用户，无疑增加了他愿意购买的欲望。

共享软件设计者一般都有一些使安装程序更加实用的技巧，下面我给出的一些原则也还是有帮助的。在最后决定使用什么样的包装之前，必须进行反复的试验比较。

打包分割化 减少盘空间占用量和下载时间的一种方法是将程序打包成几个部分。例如：主程序放在一个部分，图像在另一部分，声音在第三部分。这样用户就可以选择愿意付出的盘空间和下载时间的多少。必须保证程序没有图像和声音（或是其它决定放在单独包装中的部分）也可

以正常运行。安装程序也要提供相应的灵活性，当用户只下载了部分打包程序时应该也能安装。

帮助详细化　我们讨论一下将第 15 章建立的帮助文件放置在独立打包中的情况。方法之一是制作一个主帮助文件和一个详细的帮助文件。主帮助文件中含有对基本命令的解释，详细帮助文件中含有用户教程、宏语言描述和指令的细节说明。同样地，用户可以选择付出的盘空间和下载时间量。程序设计者也会从中受益：共享软件无法预测是否畅销，因此在没有回报时就投入大量的时间和金钱是很困难的，而用这种方法可以减少投资，直到有购买态势之后再继续投入。

技巧　已经有一些共享软件公司用模块化的方法拓展销售。例如：当用户下载时，ButtonWare 总是提供一个简单的帮助文件和程序的主要特征，购买时再允许用户下载全功能的程序和完整的程序文档。许多用户会抱怨称之为“半残的软件”，在购买之前根本无法测试！不幸的是，除非设计者能充分地说服用户购买，否则用户是轻易不会掏钱的（我说的是实话，无数共享软件公司都因为销售不畅而倒闭了，但它们的软件却仍在广泛使用）。

商业软件

我并不打算假惺惺地要教微软之类的大公司如何包装产品、开发市场——它们的市场开拓效果已经是有目共睹，我似乎在班门弄斧。商业软件区别于共

享软件的一个特点是软件的大小。大的软件公司通常集中于一个或多个产品，有许多专业的设计小组完成整合封装。研究一下这些软件，也会使我们这些普通的程序员从中受益。

过去的几个月中，每当我安装一个成熟的软件产品时我都非常留意。装过30个产品之后，我审视了一下这些记录，结果令人吃惊。例如：前面我说过，对于共享软件的作者，设计一个好的安装程序会有助于销售，因为他财力有限，必须用第一时间抓住用户。令人惊奇的是商业软件也情同此理，只不过方法不同罢了。用户已经购买了这种产品正在安装，但是还有其它增值产品呢？商业软件厂家总是利用安装的时机，推销相关连的增值产品（比如安装字处理程序时，介绍字典软件）。

另外，我发现商业软件在安装时总是告诉用户程序的新增特性。这倒不奇怪，因为这些公司早就注意到用户不留意 README 文件，安装往往就是用户装载一个软件的最后一步。同样地，他们要吸引用户，因此在安装程序中就要使用户对程序有一个整体概念。

商业软件总是带有许多多媒体功能也就不足为奇了，因为这些公司有足够的开发实力；即使不像这些大公司那样实力强劲，同样应该注意我们强调的该作什么，不该作什么。在安装程序中提供的版本虽然小但要代表原貌才能吸引用户，牢记你要作的就是尽可能给用户最好的第一印象，因此一个精彩的安装程序不仅仅是应用程序的良好开端，也是用户了解程序的窗口，它很可能会决定用户是否来购买你的程序。

你也能够从商业程序中吸取一些教训。其中之一是安装程序太复杂，以至

于测试得不完善，最近，我就遇到了这样一个产品，安装时发现它的帮助屏幕是给老版本做的——厂家忘了更新它了！结果呢？因为该产品也没带任何印刷的说明书，我就很困惑产品中的各部件有什么特性了。如果厂家认真完整地测试了安装程序，我就会明确地得到想要的信息，以决定安装哪些部件。

另一个很奇怪的问题是产品根本安装不上。似乎是因为安装程序中用了某种特殊的图像，碰巧我的机器不支持。换了台机器，安装后该程序运行得很好，但安装程序本身还是显示不全。同样地，厂家为了追求精彩，却适得其反了。

从这一节可以看出，研究一下别人封装产品的方法，能取得事半功倍的效果。我发现事先写下注意事项，在动手编写安装程序之前仔细阅读一遍，可以少犯许多错误（几乎没有人可以一蹴而就）。

16.2 收 集 文 件

千万不要以为不经过研究，就可以在二、三秒内作好一个安装程序，多数情况下，Windows 环境中的所有程序都比程序员预计的情况要复杂得多。例如：本书中的所有例子都依赖于 C 运行时库文件，到目前为止我还没有提到在应用程序中加入这些文件，源代码中也没有对它们的引用，这是因为第一次安装 Visual C++时，已自动放在 SYSTEM 目录下，关联是自动进行的，在设计应用程序时不需要了解。对 MFC 文件和数据库应用涉及的文件情况相似，Visual C++即使没有完全地却也基本上帮你增加了所需的文件。然而，现在要做的是将应用程序完整打包提交给最终用户，这时你再也不能无视这些文件的存在了——

它们也是应用程序的一部分，必须放在打包之中。问题是如何找出应该包含哪些文件呢？

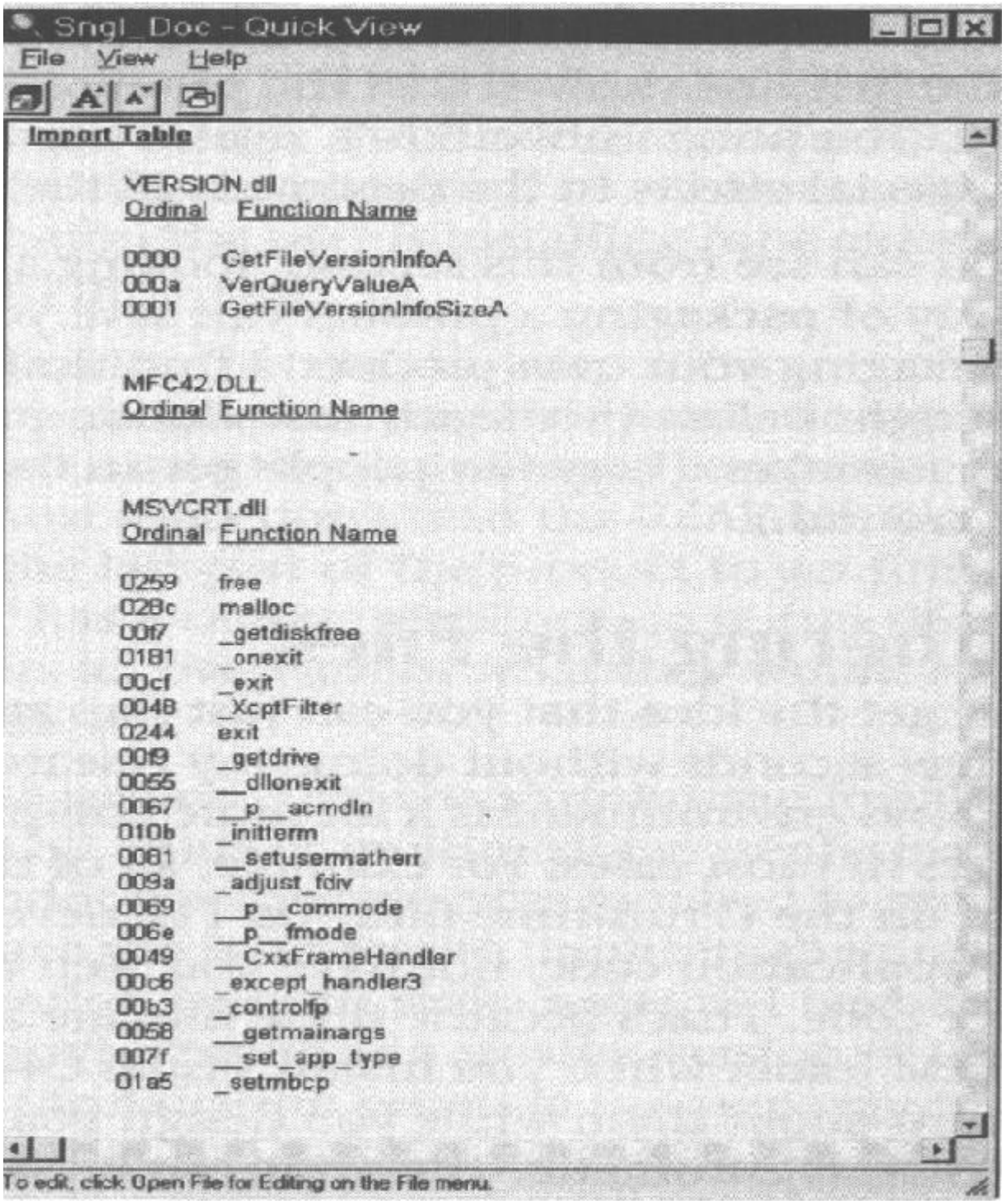
找出应用程序应该包含的所有 DLL 非常费时间，特别是当使用测试错误的处理方法时更是如此。然而，你可以使用如下的三种方法来进行测试错误处理：

方法一　使用 Windows QuickView 程序查看应用程序的 Import Table 表项，Import Table 列表列出了应用程序所有用到的 DLL。下页图（上）是我们在第 2、3、15 章介绍过的 Sngl_Doc 程序的 QuickView 程序列表，它不仅显示了该应用程序用到了哪些文件，而且给出了其中的函数。

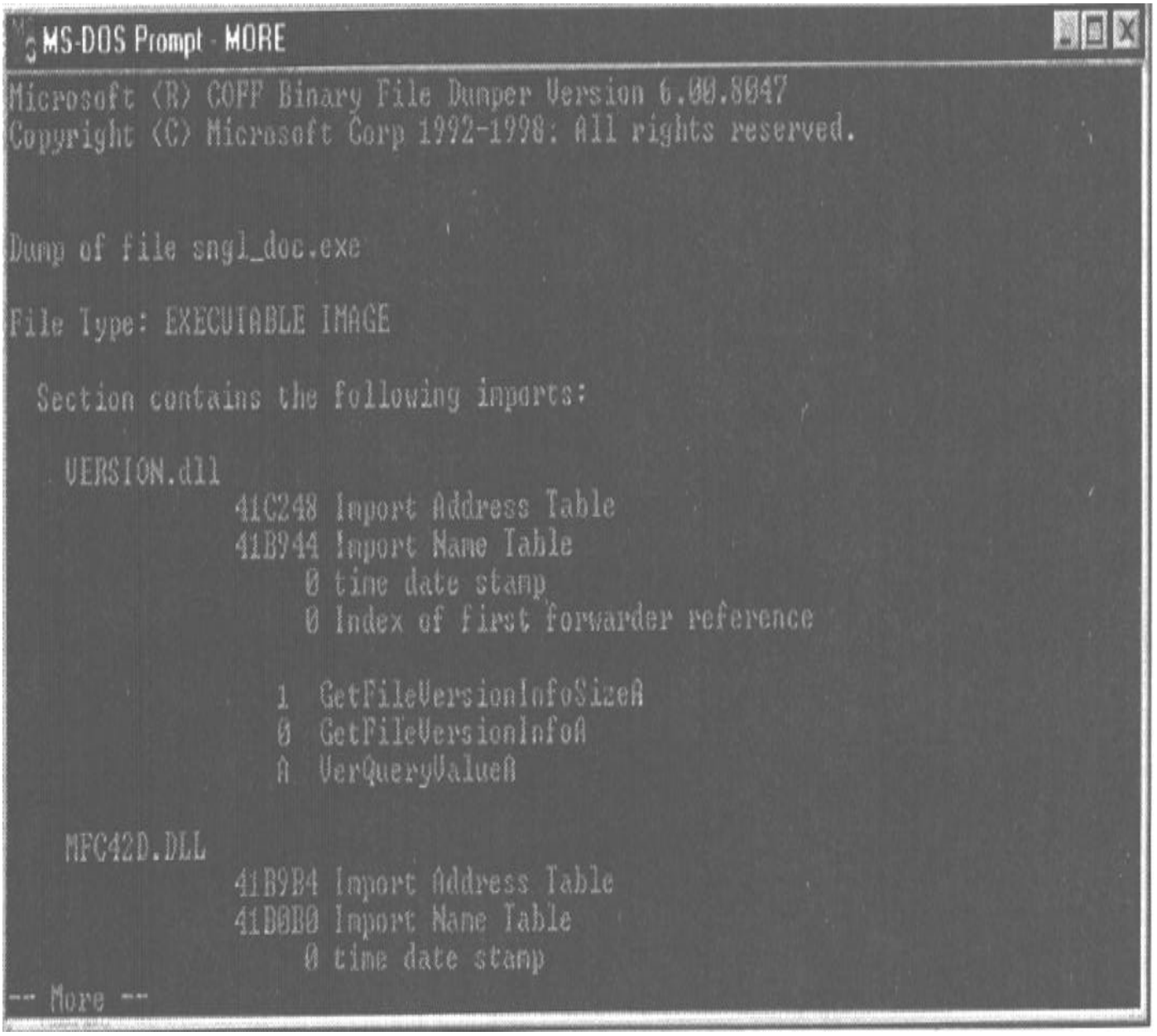
方法二　如果程序需要可以使用 DumpBin 程序给出 DOS 命令行方式显示的程序列表。用起来略微有点儿麻烦，但却常常可以发现 QuickView 发现不了的文件。而且它提供了更多的选项，可限定查找信息的种类，输入命令格式为：DUMPBIN /IMPORTS SNGL_DOC.EXE。下页图（下）是输出结果（这是个命令行程序）。

技巧　使用 DumpBin 实用程序时可能会遇到系统报出缺少文件的信息，如果使用的是安装时的缺省设置，那么 MSDIS100.DLL 和 MSPDB50.DLL 文件都在 Program Files/DevStudio/SharedIDE/bin 目录下，在 DumpBin 的执行过程中需要用到它和 VC/bin 目录下的 LINK.EXE 文件。

方法三　如果有最新版的 Windows SDK，就会看到 Depends 实用程序，用它就可以找出程序中用到的 DLL。


Sngl_Doc - Quick View
File View Help
Import Table
VERSION.dll
Ordinal Function Name
0000 GetFileVersionInfoA
000a VerQueryValueA
0001 GetFileVersionInfoSizeA
MFC42.DLL
Ordinal Function Name
MSVCRT.dll
Ordinal Function Name
0259 free
028c malloc
00f7 _getdiskfree
0181 _onexit
00cf _exit
0048 _XcptFilter
0244 exit
00f9 _getdrive
0055 __dllonexit
0067 __p__acmdln
010b _initterm
0081 __setusermatherr
009a _adjust_fdiv
0069 __p__commode
006e __p__fmode
0049 __CxxFrameHandler
00c6 _except_handler3
00b3 _controlfp
0058 __getmainargs
007f __set_app_type
01a5 _setmbcp
To edit, click Open File for Editing on the File menu.

找出这样一个文件清单还不是文件收集过程的结束，还需要找出相关的全部 DLL 文件。显然像 USER.EXE 文件这样安装 Windows 时就具有的文件不用包含进去，但必须找出那些没有的文件。总而言之，必须手工地检查每个文件及其相关文件，确保都包含在应用程序的打包中。



技巧 先仔细审查一下只装过 Windows 的机器中有哪些文件，如果以后要建立多个安装程序，这个文件清单就很有用了。

当应用程序文件清单完整之后，把它们放在一起，拷贝到装有更高版本的 Windows 的机器中，检测是否兼容。当缺少文件或是文件不在 SYSTEM 目录下时，就会看到警告信息。一定要保证放在 SYSTEM 目录下的文件必须到位，因为将来用户使用起来也是这样。

注释 尽量将全部 DLL 文件放在应用程序目录下，而不是在 SYSTEM 目录下，这样有助于删除程序的实现。而且会少受用户安装的其它程序的干扰，例如：所用到的某个 DLL 在其它程序中也用到了，而且是个老版本的，安装时会把你的新版本 DLL 给覆盖了。

16.3 建立安装程序

现在你已经清楚如何打包应用程序了，包括决定采用何种模式，比如：公司内部应用软件、共享软件和商业软件。 还应该有一张所用到的全部文件的清单，并保证包括了每个文件安装所涉及的方方面面，例如：写出必须安装在 SYSTEM 目录下的文件。最后，应该在一台只装有 Windows 的机器上测试这些文件，直到通过之后，才能开始编写安装程序。

注释 本节内容以 Visual C++6.0 的 InstallShield 为基础，老版本的 Visual C++中的 InstallShield 版本太低，使用时必须修改某些过程

才能满足需求，即使你所使用的 InstallShield 与本书所用的 InstallShield 版本相同，你所看到的屏幕显示和选项也会因所封装的程序不同而有所差异。简而言之，使用时看到的和叙述的有些差异无关大局。但必须安装 InstallShield 产品，它在 Visual C++ 6.0 CD 的 Ishield 目录下。我们假定你已装好了。

到目前为止，我们已向第 2 章建立的应用程序 Sngl_Doc 中加入了一个资源文件（第 3 章）和一个帮助文件（第 15 章），可以进行打包提交给用户了，它至少可以作为一个例子。下文给出了建立一个典型安装程序的步骤，这些没什么费解的，就是为了程序的发布。如果非要从中提取出一个模型，近似于我们前面提到的企业内部应用程序型的打包情况。

注 本节讲述完整的安装程序，但实际应用中必须进行修改以满足特定的程序和安装环境的需要。

1. 启动 InstallShield 6 Free Edition 程序。你会看到如图 16.1 所示的窗口，注意：事先并没有定义工程文件，但在 Projects 窗口中出现了一个安装向导，而且看上去不像 Visual C++中的标准窗口，但随着工作的进行，你会发现它非常有助于安装程序的建立。同时请注意一下右下角的 InstallShield 链接，单击将打开浏览器，打开 Internet 上的 InstallShield 在线帮助站点。

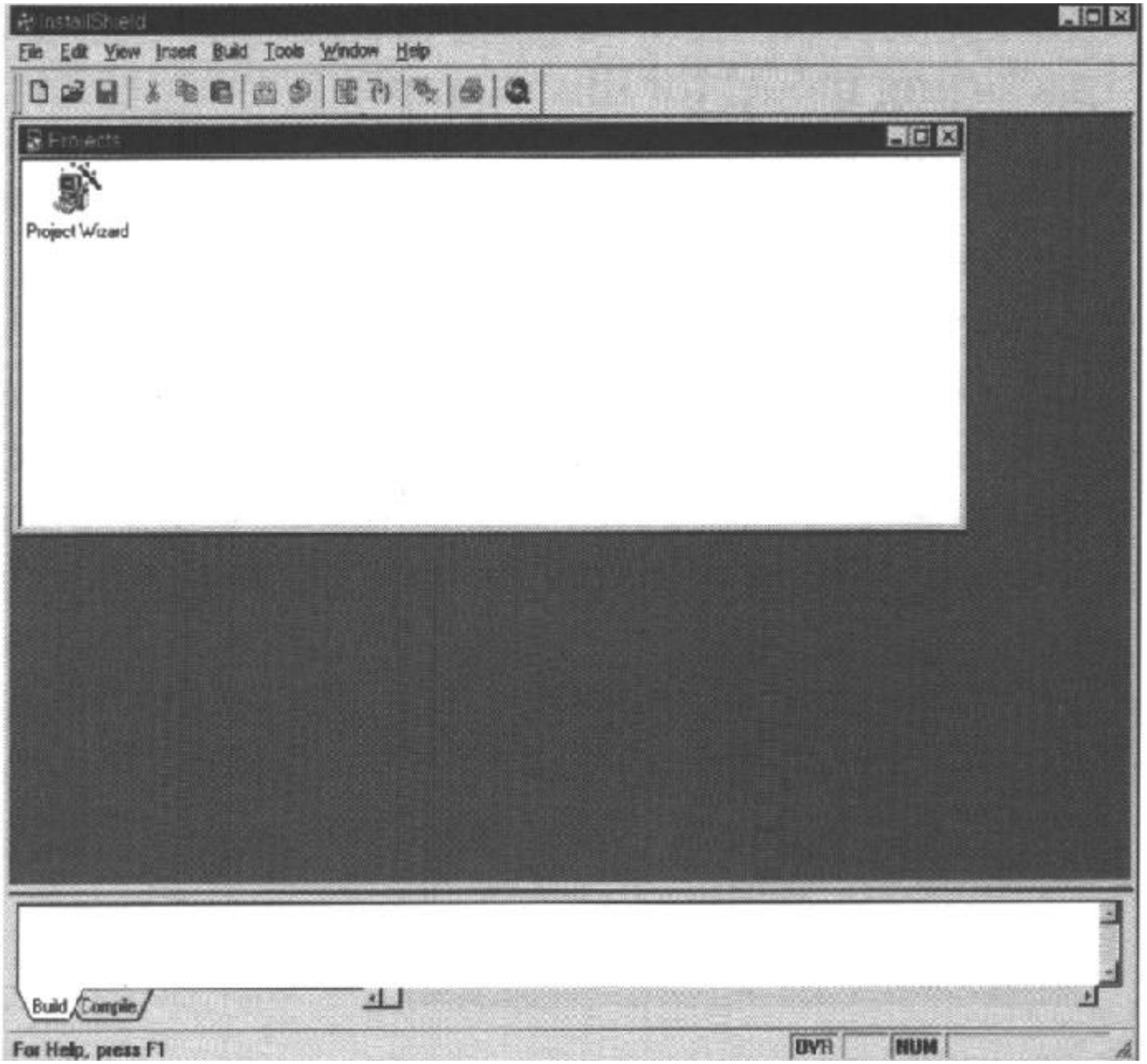
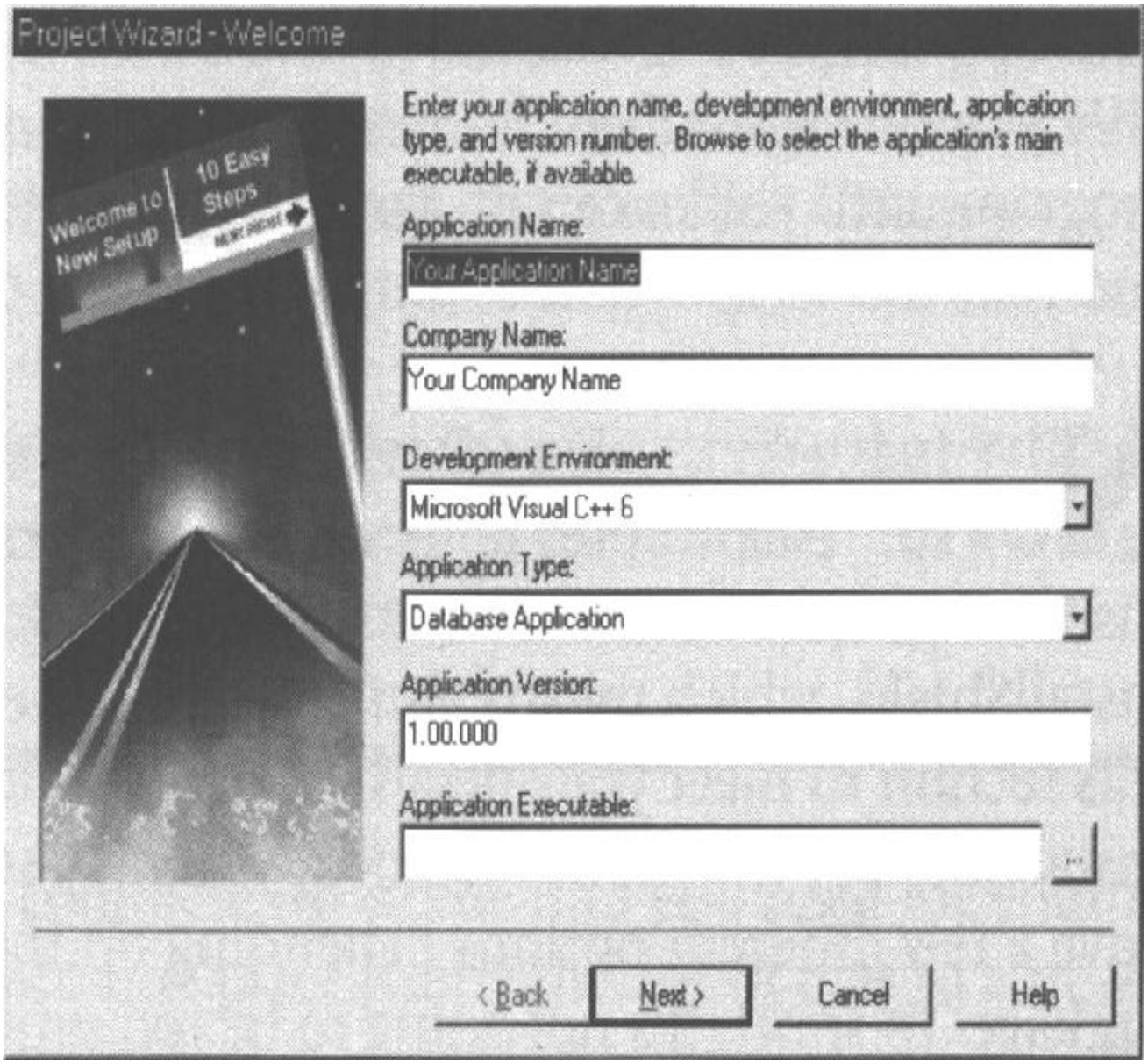


图 16.1　InstallShield **让你使用与** Visual C++**中创建应用程序所用技术相似的技术创建安装程序**

2. **双击** Projects **窗口的** Project Wizard **图标，你会看到如下图所示的** Project

Wizard-Welcome **对话框：**



注 InstallShield 的 Professional Edition 在 Welcome（及其它）页提供了高级特性——请一定要读一读帮助文件的增强特性部分。

3. **输入应用程序的名称，示例程序使用了** Single Document Application

Example。

4. 输入公司名称，示例程序用的是 A Sample Company。

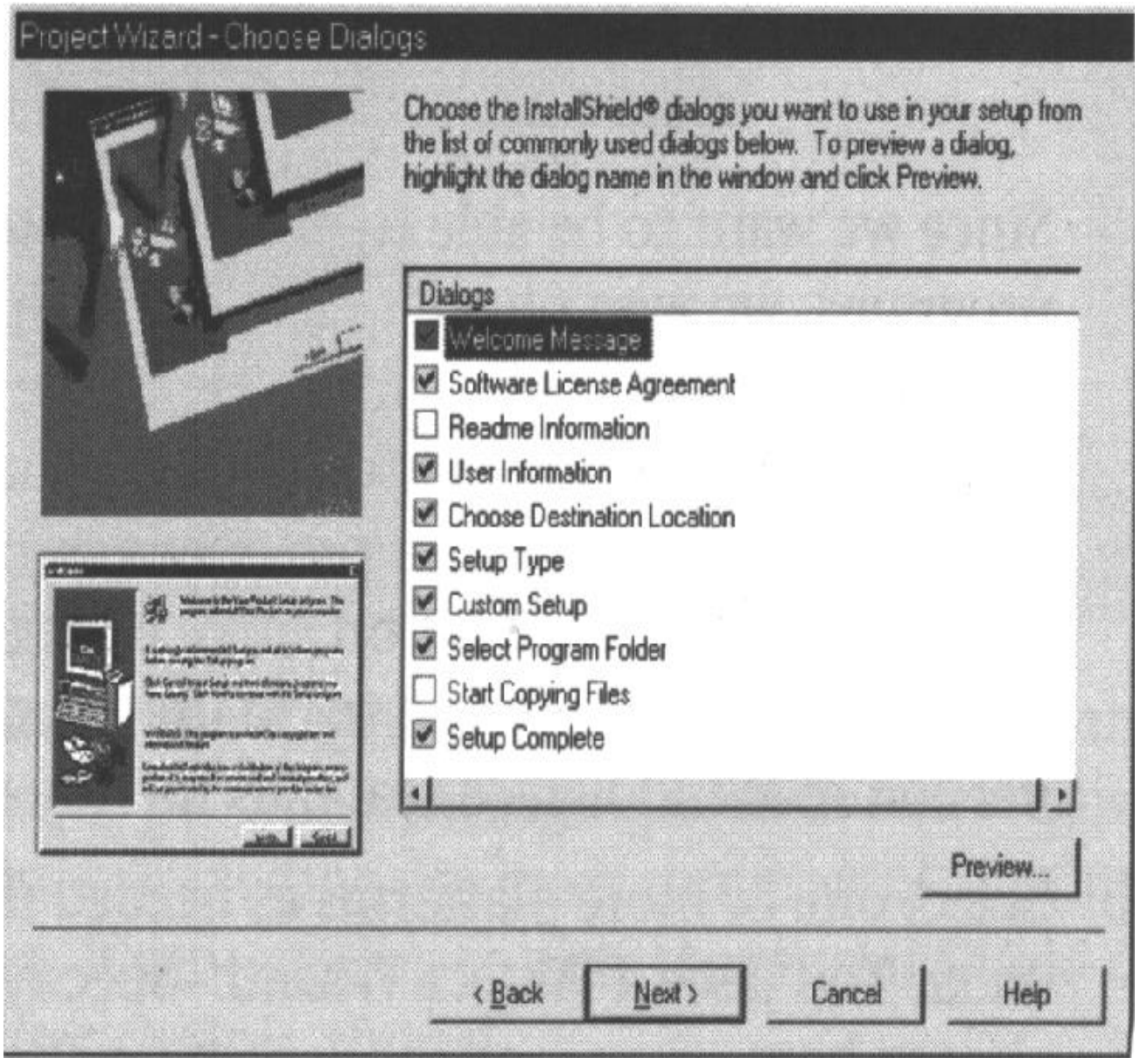
注　InstallShield的Free Edition(免费版)只为 Development Enviroment 域提供了一个选择项：Microsoft Visual C++ 6。

5. 在 Application Type 域选择一个列表项，示例程序选的是 Software Development Applicaion；该安装程序没有标准程序类型，而 Software Development Applicaion 支持实用程序。

6. 在 Application Version 域输入版本号，我们用的是 1.0，当然可以沿用自己公司中的编号。

7. 单击 Application Executable 域旁边的“...”（省略号）按钮，你会看到一个标准打开对话框用于查找硬盘上的应用程序，示例程序用的是 SNGL_DOC.EXE（我们在第 3 章中建立、第 15 章中修改的那个程序，在 Resource 文件夹下）。

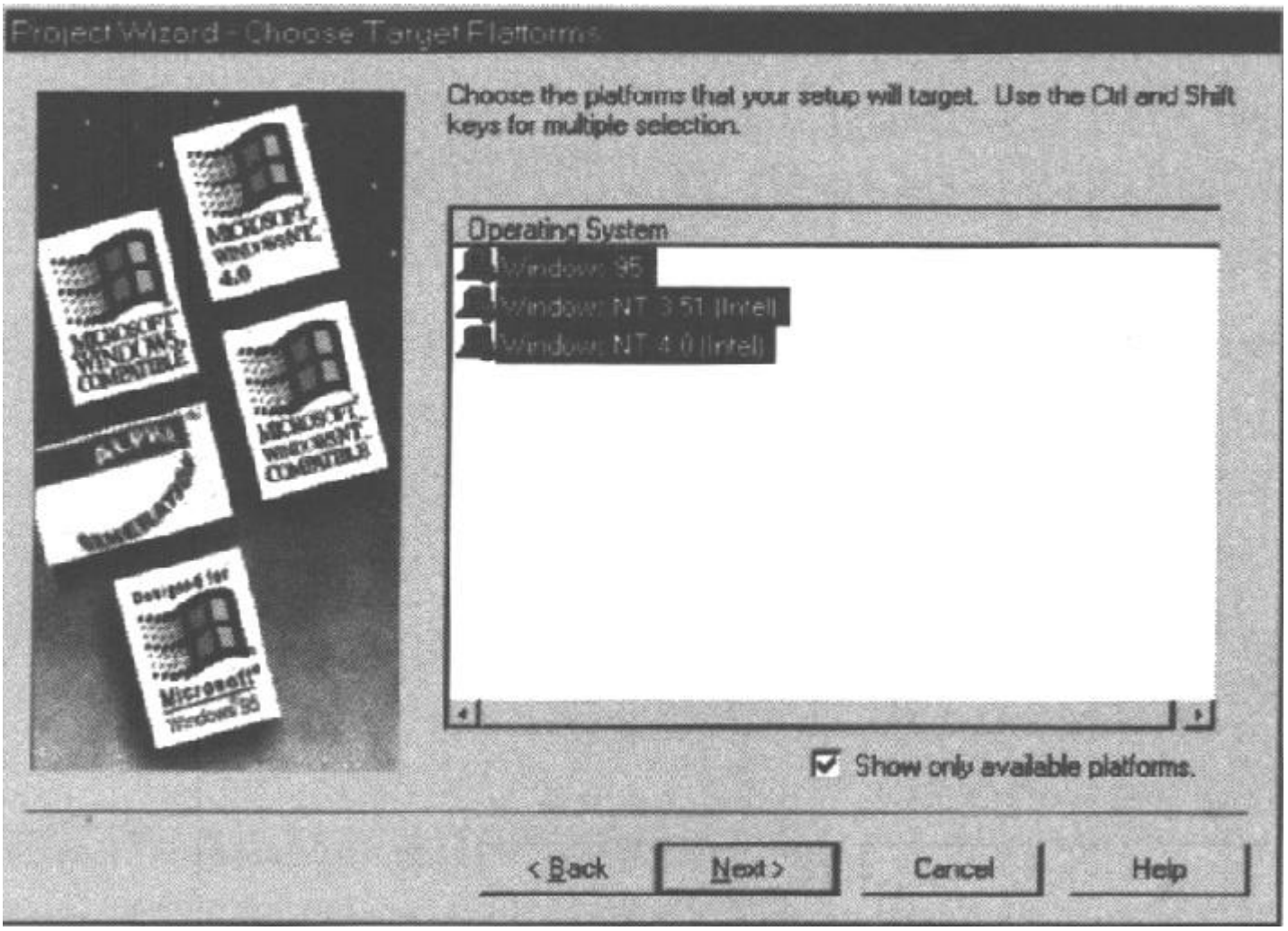
8. 单击 Next，你会看到如下图所示的 Project Wizard - Choose Dialogs 对话框：



在这里选择的一系列对话框会在用户的安装过程中显示出来，因为我们使用的是企业内部应用型的程序，所以不选中 Software License Agreement、Setup Type **和** Custom Setup **对话框。对本章前面部分讨论过的其它打包模型来说，则可能需要选择其中的某些项目。在这个对话框中有几个需要注意的地方：首**

先，当从一个对话框选项转移到另一个对话框选项时，请注意该对话框左下角的变化，Project Wizard 会显示出该对话框的一个微型缩影，以方便你确认是否要选用这个对话框；加亮一个对话框后单击 Preview 按钮，所看到的对话框与用户看到对话框相同。显然，你以后还可以修改这个对话框，但预览方法有助于你预先了解安装程序中要使用的对话框。

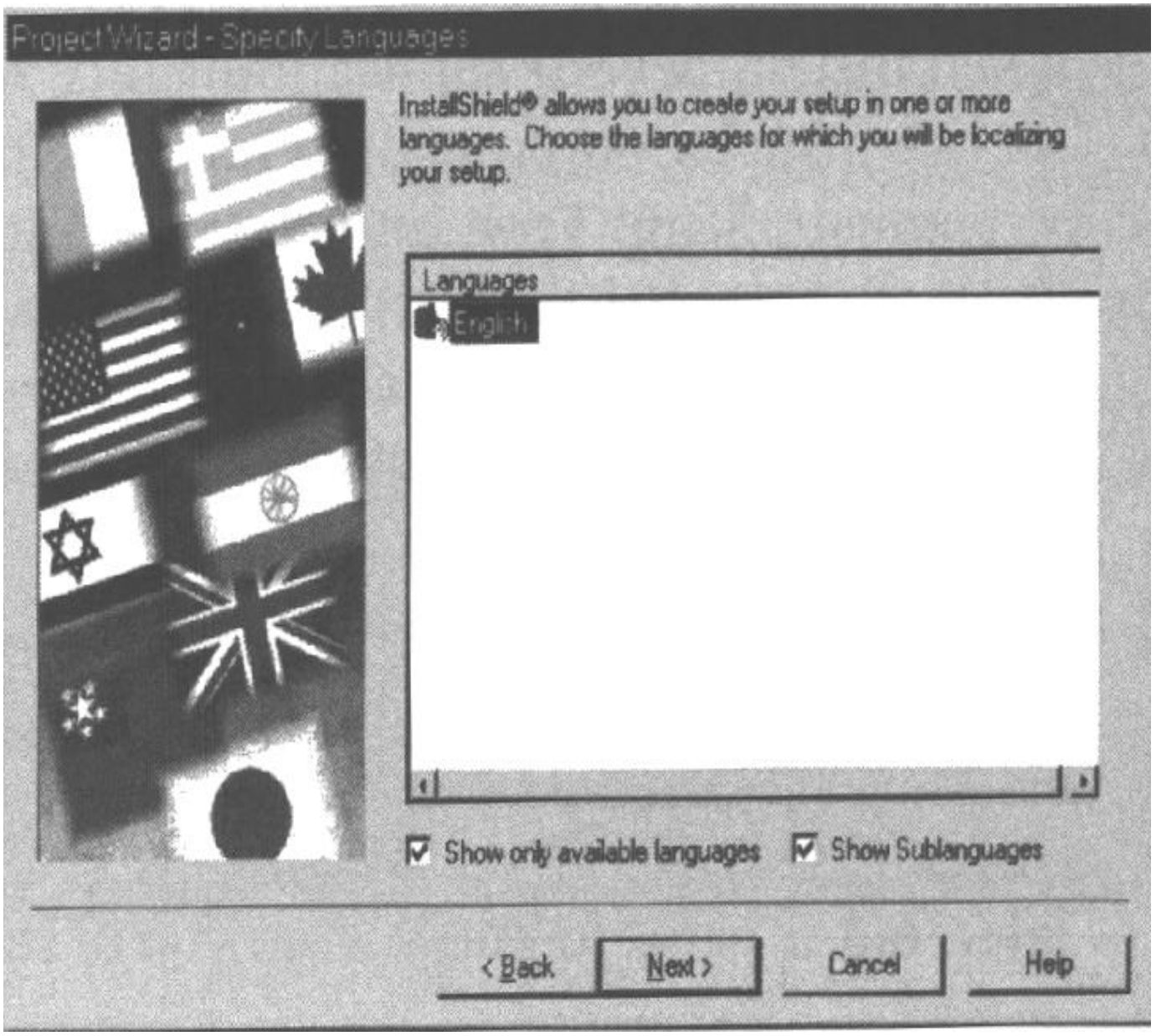
9. 选择想要加进安装程序的对话框，单击 Next，你会看到如下图所示的 Project Wizard - Choose Target Platforms 对话框：



由于我们想把我们的程序安装到各种支持程序上，因此不需要改动要安装的平台。然而，在某些情况下你可能想作些选择。减少支持平台的种类会同时带来两种结果：第一、减少了所建立的安装程序的大小——这对于开发占用空间少的共享软件很有帮助；第二、减少了用户在不适用的 Windows 版本中使用程序的可能性——这对所有程序设计者都有帮助。

注　不选中 Show only Available platforms 复选框时，系统将会显示专业版 InstallShield 提供的所有平台选项。

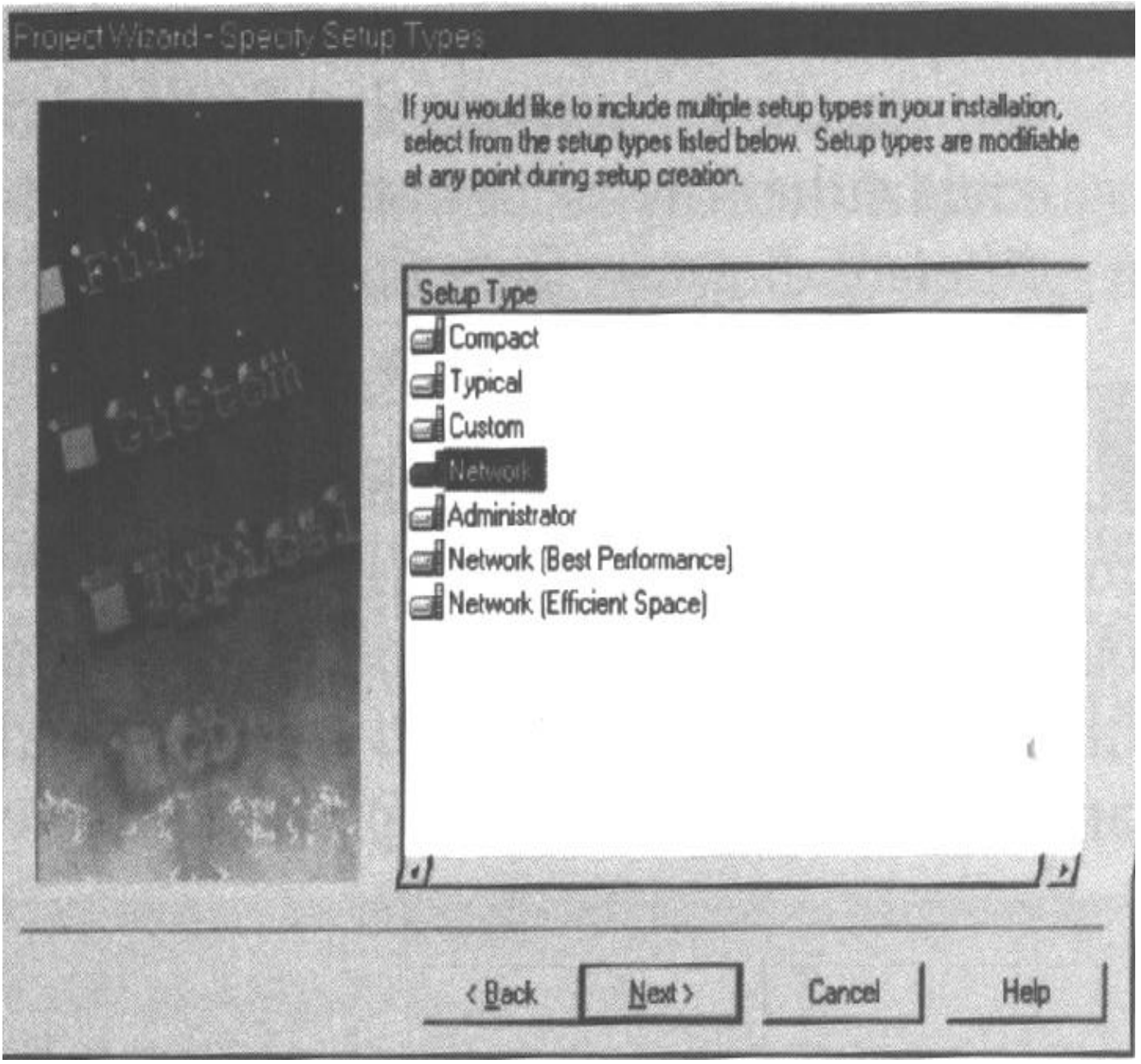
10. 根据需要选择一种或多种（至少一种）平台，单击 Next，你会看到如下图所示的 Project Wizard - Specify Languages 对话框：



由于 Free Edition **版只支持一种语言，所以这里不需要作改动。与所有其它的选择一样，多选并没有必要。增加语言必然增长安装程序的长度，也会给用户造成困惑，最好只选所用的语言。**

11. **选择所用的一种或多种（至少一种）语言，单击** Next**，你会看到如下**

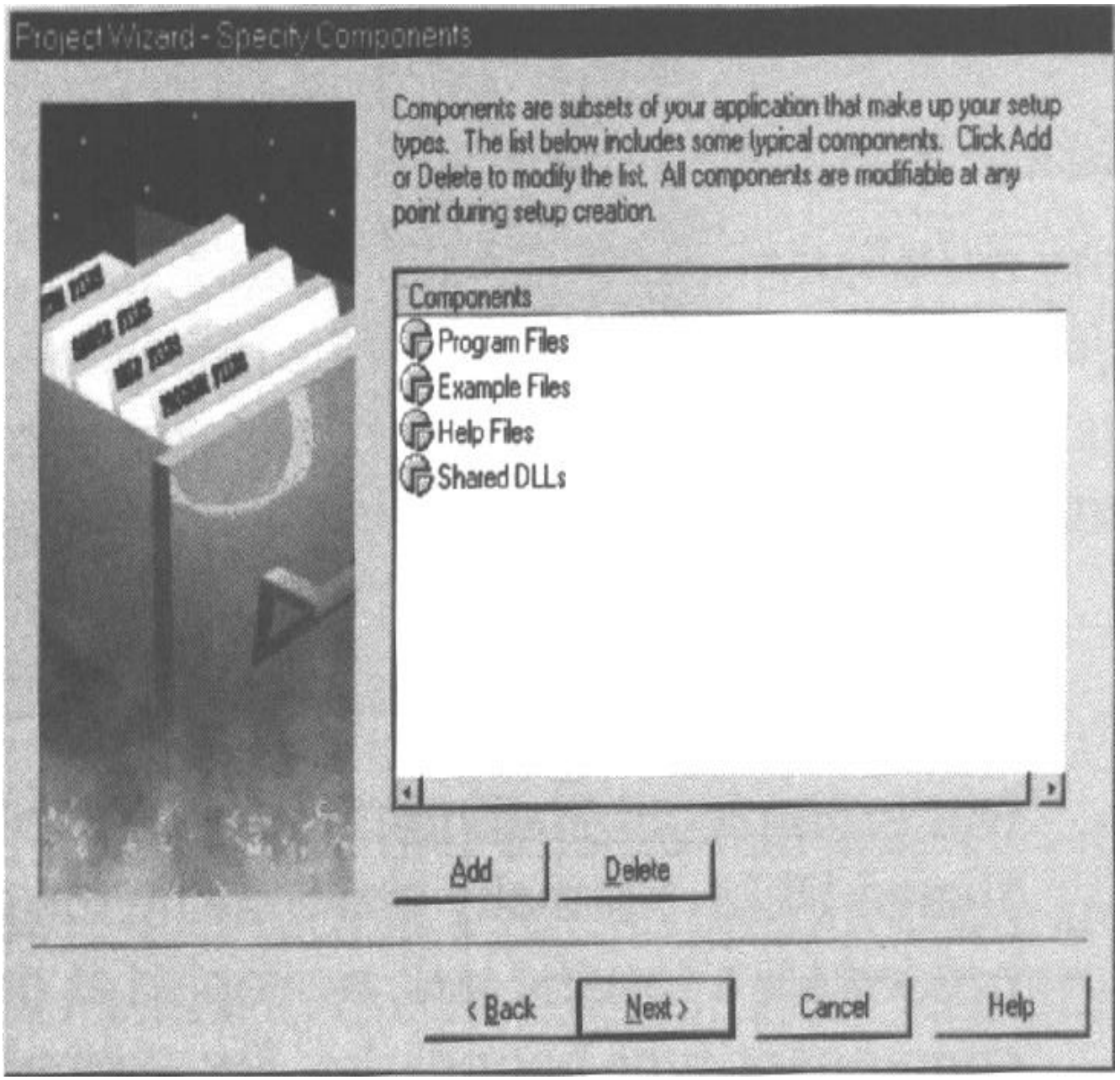
图所示的 Project Wizard - Specify Setup Types **对话框：**



因为我们使用的是企业打包模型，所以我选择了 Network **安装类型.其它打包模型要求作出其它选择，包括常见的方式：定制方式、典型方式和最小方式。**

有趣的是 InstallShield 还提供了一些其它的选择。

12. 选择一种或多种安装类型，然后单击 Next，你会看到如下图所示的 Project Wizard - Specify Components 对话框：

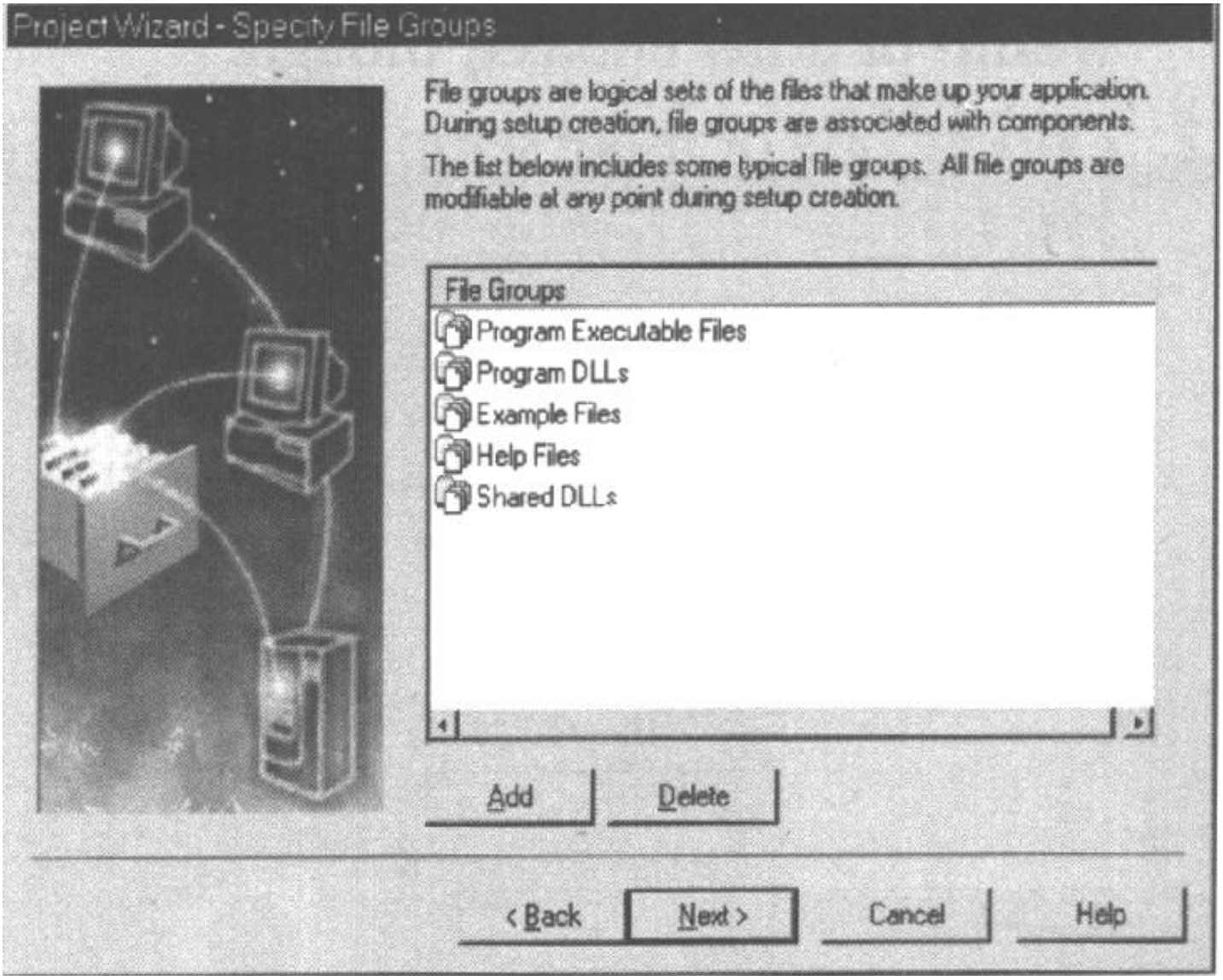


Project Wizard 并不知道你打算安装哪些组件。示例程序使用了 Program Files、Shared DLLs 和 Help Files。你需要每次选择一个组件类型，并定义该组件类型下的文件。组件类型对程序进行了主要部分的划分，例如，某个用户在安装时希望安装所有的示例文件，但不安装帮助文件。组件类型并不需要指出具体由哪些文件完成该组件的任务，而是让用户选择把不同组件中的文件安装在同一目录或不同的目录下。

技巧 可以根据需要增加新的组件类型。例如：如果是数据库应用程序需要加入 Database File 组件，作起来很简单：单击 Add 按钮，向导会自动加入新的列表项，输入组件名称，按 ENTER 键就行了。

13. 选取 Example Files，单击 Delete，Project Wizard 会删除该组件类型。

14. 根据需要增加、删除组件类型，单击 Next，你会看到如下图所示的 Project Wizard - Specify File Groups 对话框：

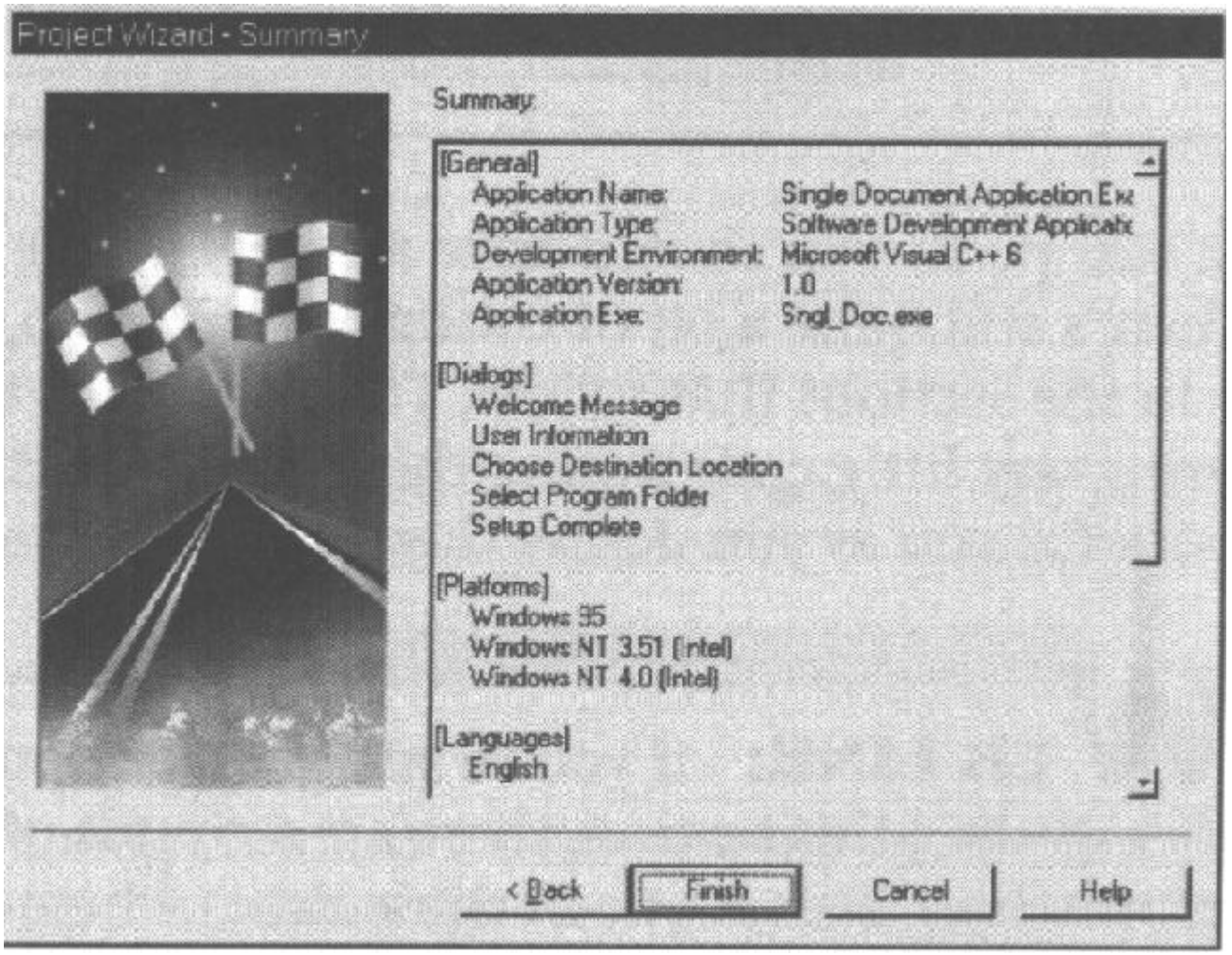


示例程序使用了 Program Executable Files、 Help Files **和** Shared DLLs。**正常情况下，你会创建一些文件组，以便完成特定任务所需的所有文件能一次拷贝到目的地。文件组不受组件类型的限制，例如：拼写检查和语法检查都依赖于同样的** DLL **文件集，但使用的规则不同，你可以将** DLL **定义为一个共用的文件组，规则文件分别定义为另外两个组。如果用户选择了拼写检查，**InstallShield

就会拷贝相应的规则组和通用 DLL 文件组，但不会复制语法检查文件组。

15. 选取 Program DLLs，单击 Delete，选取 Examples Files，然后单击 Delete，删除本示例程序中不需要的两个文件组，显然文件组的选择取决于程序的组织方式、所需文件的复制位置。请谨记，同一文件组中的所有文件都将拷贝到硬盘的同一个目的目录下。

16. 根据需要增加、删除文件组类型，单击 Next，你会看到如下图所示的 Project Wizard - Summary 对话框：



这时应检查所列清单，保证正确无误，然后再开始生成安装程序。

17. 单击 Finish。InstallShield 用所选定的选项生成安装程序，界面会变为如图 16.2 所示的样子，注意这时你会看到创建安装程序所需的 C++源代码，你可以像修改其它工程那样对这个工程进行修改。

注释 不要以为 InstallShield 生成的源代码文件与普通工程生成的源代码文件完全一样，实际上，它生成的文件中有些是你从未见过的辅助性文件，比如安装规则文件（SETUP.RUL），如图 16.2 所示。

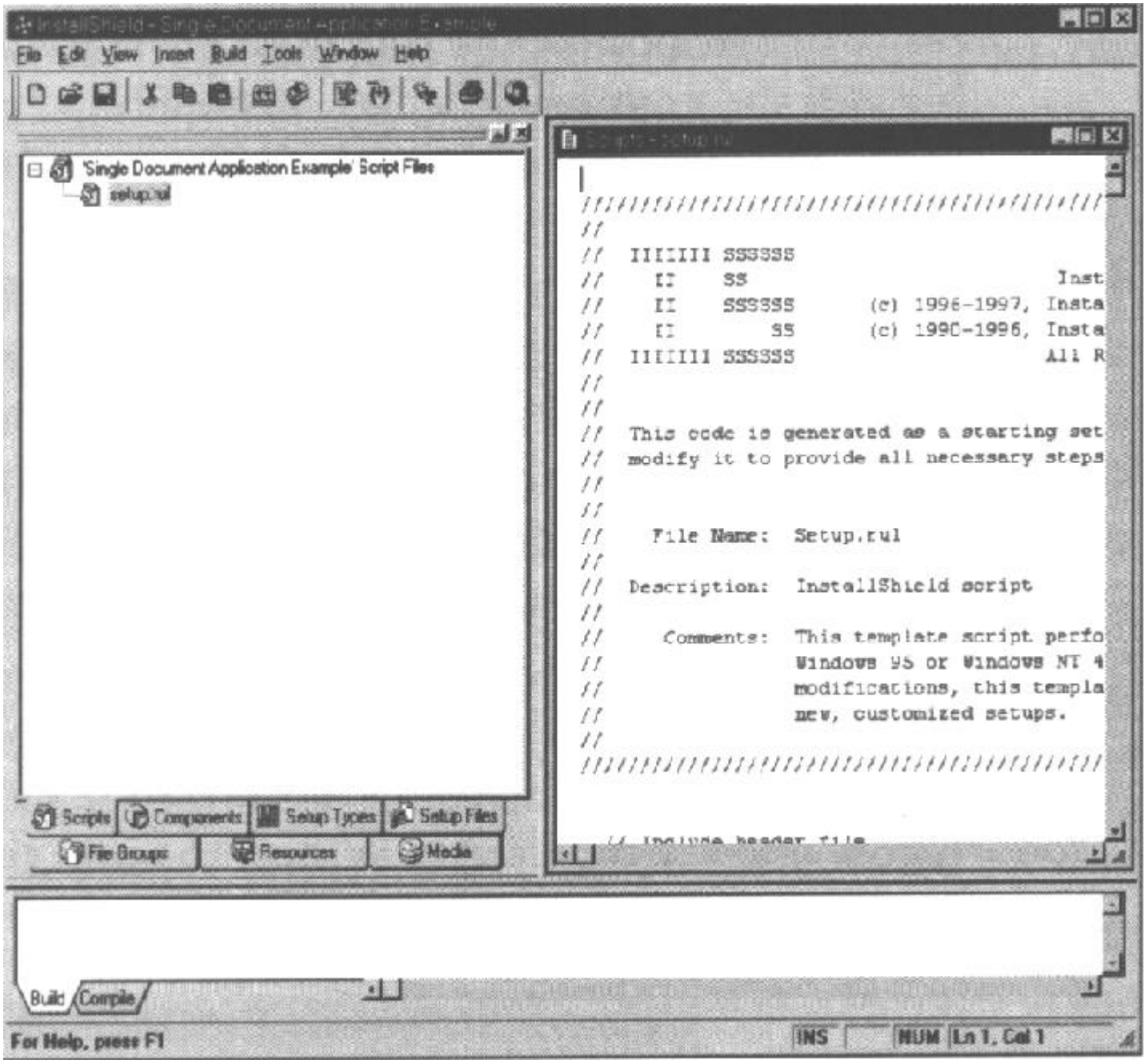


图 16.2　**建立了工程文件之后，**InstallShield **会自动生成并显示结果源代码**

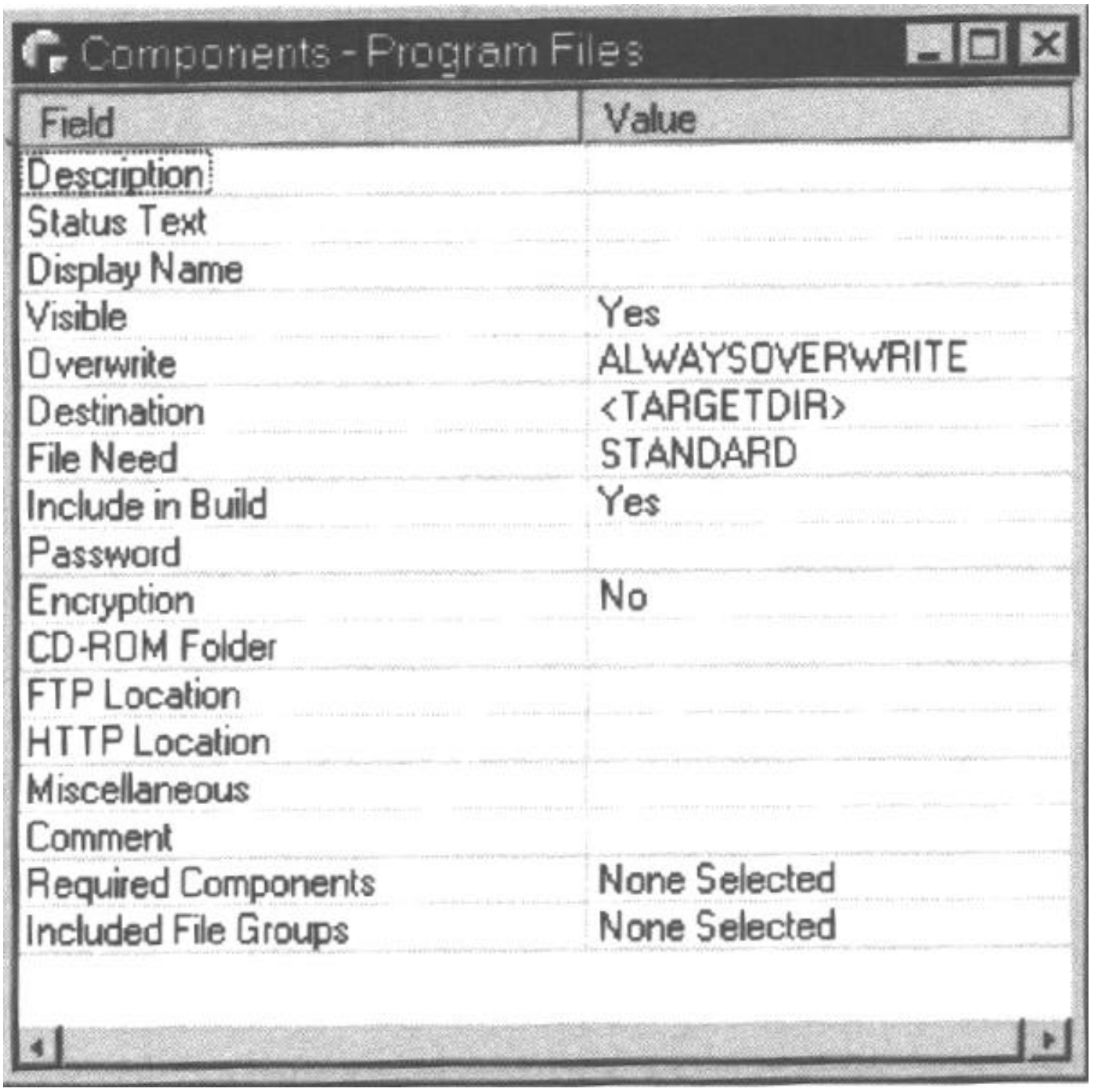
这时，就有了一个空白的安装程序外壳，还需要作很多的配置，我们将在下节阐述。最重要的是要看到整个过程中几乎没有写什么代码，在定义了各种安装程序元素之后 InstallShield 就替你实现了。

Web 链接 如果使用 InstallShield 还有问题，有许多方法可以得到帮助。这时最需要读一下 README 文件，其中含有许多 Internet 地址，例如：新闻组站点 http://support.installshield.com/newsgroups/default.asp，它链接到了几个 InstallShield 专题的新闻组，只需单击一下，在你的新闻阅读器中就会建立一个新的文件夹。甚至对 Free Edition 版也有专门的新闻组，地址为 installshield.is5.free-edtion。还可以找到讨论 IDE、脚本和多媒体用法的新闻组，总而言之，这些新闻组提供了与其它程序开发者交流使用打包程序、成功打包应用程序的经验。

设置组件

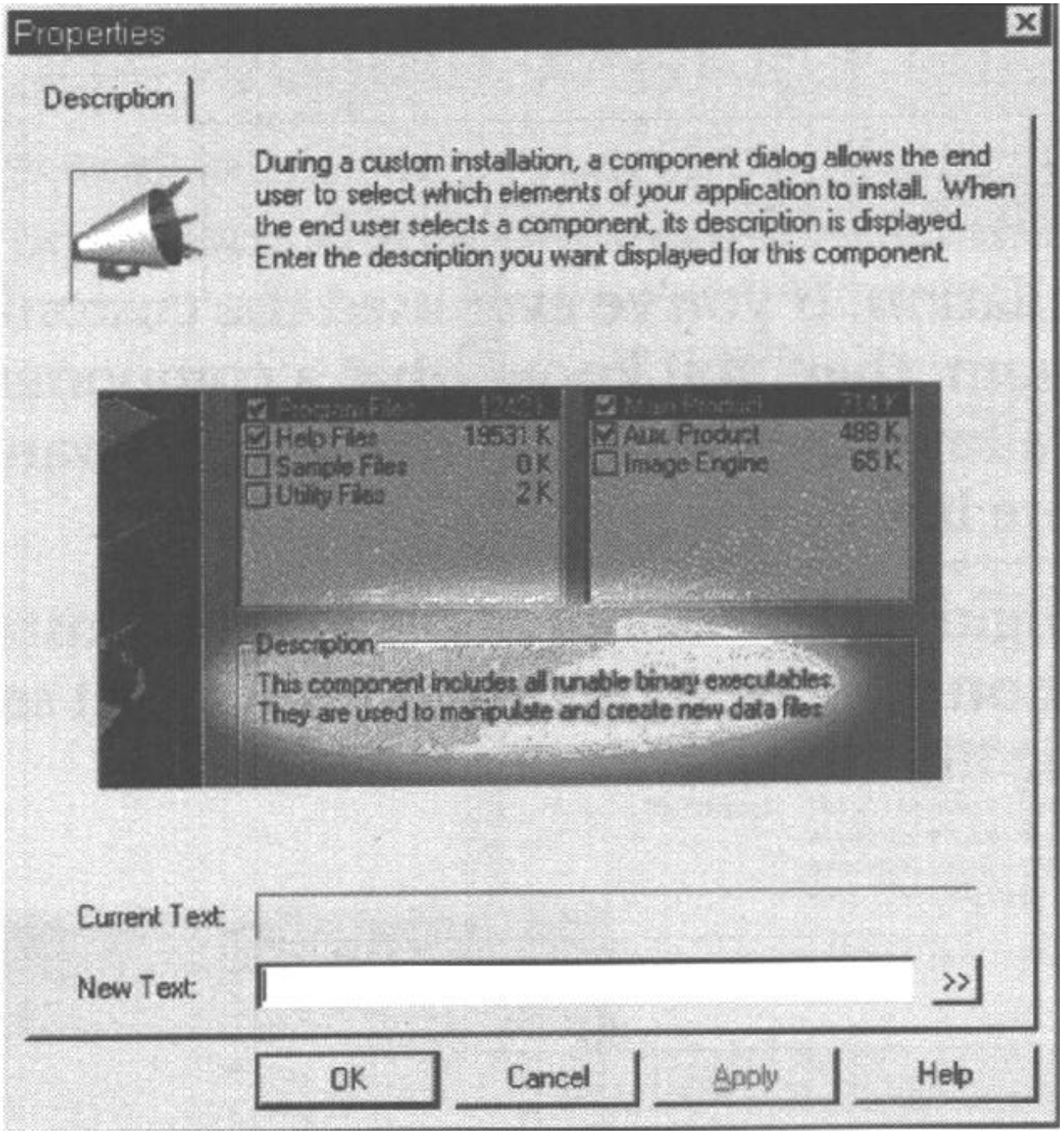
首要的任务是设置各种组件。组件是安装过程中出现的选择项，如果以前用过定制安装，就会很清楚。它们是一系列的复选框，这些复选框让你选择是否安装某个程序项。

配置组件相对比较容易，单击 Components 选项卡，你会看到如下图所示的 Components - Program Files 对话框：



组件对话框中包含了所选组件的完整清单，这里有三个组件：Program Files、

Help Files、 Shared DLLs。**在阅读下一节之前，必须全部设置这三个组件。双击** Description **属性，你会看到如下图所示的** Properties **对话框：**



其它所有的属性对话框与该对话框相似。每个属性对话框都用来描述要用该属性完成什么任务。也可以输入属性的值，本例中输入为：All the files required to run the application。完成后单击 OK，Description 属性就包含了所输入的内容。

不需要改变所有的组件属性——真正修改你进行的修改取决于所建立的应用程序的打包类型，例如，对于公司内部应用程序，一般不允许用户选择要安装的组件，所以没必要定义组件描述。但是，无论选择何种打包模型，有些属性都需要修改，下面我们逐一讨论：

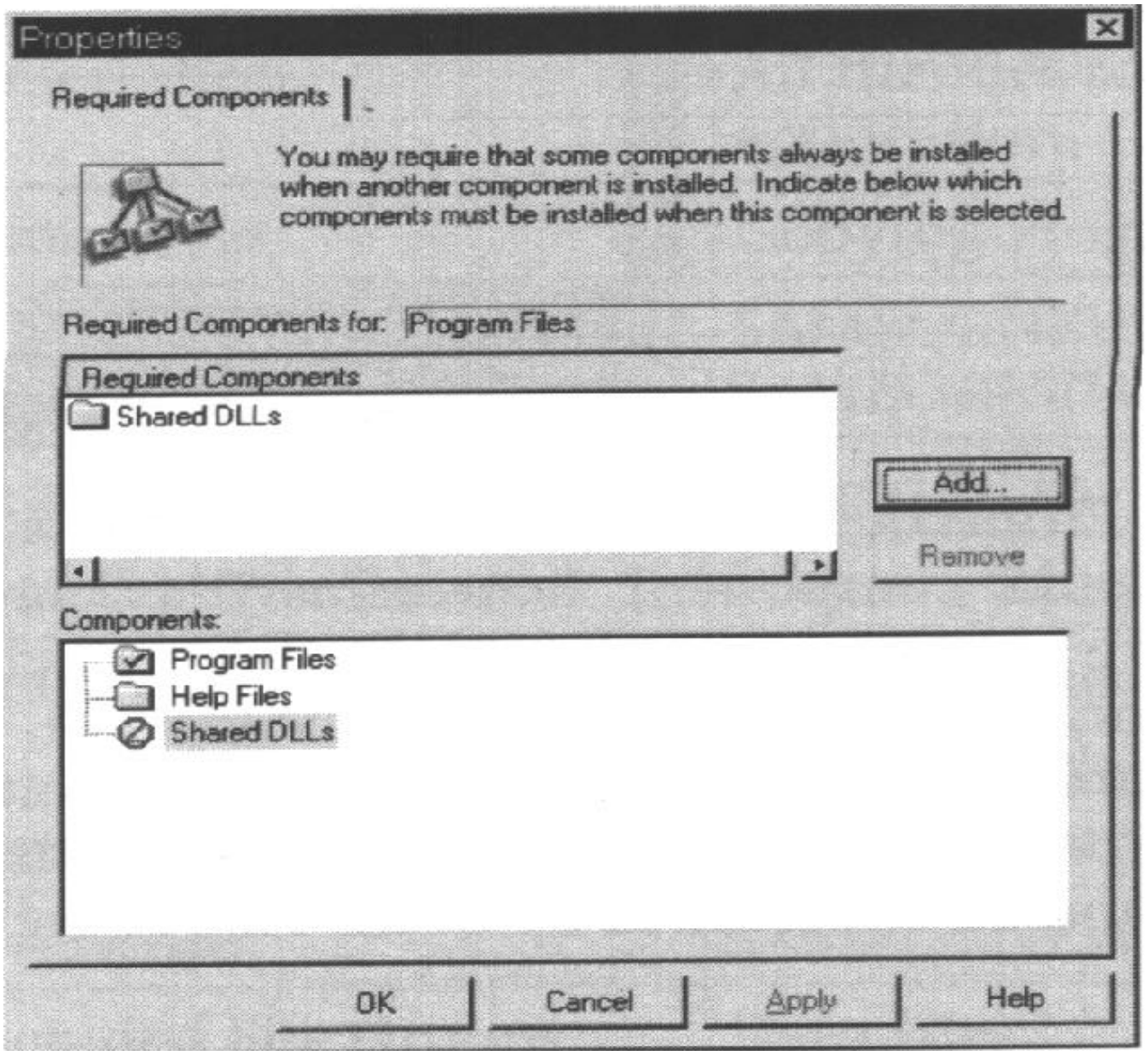
Status Text（状态文本） 这些文字是在安装程序向目标盘拷贝文件时用户看到的提示。Progress 对话框将显示一些类似于“ Copying program files... ”的提示，这些提示适用于缺省组件。对定制组件来说，你或许想给出一些特殊信息。

Installation（安装） 如果遇到过自己的最新版本的 DLLs 被其它程序覆盖的情况，你就会切身体会到这个属性的重要性。该选项告诉 InstallShield 在覆盖文件之前你是否想先检查它的时间戳，我总是使用 NEWERVERSION/NEWERDATE 或 SAMEORNEWERVERSION /SAMEORNEWERDATE 选项来代替缺省的 ALWAYSOVERWRITE 规则。实际上，这些选项告诉 InstallShield 仅当所装文件比硬盘中现有文件日期更新或版本更高时才覆盖原文件。

Destination（目的） 所有应用程序文件的标准安装位置是目的目录，它由用户选择。然而有些情况下使用用户设定的目录会浪费盘空间，例如：大多数 Visual C++程序运行时要求 C 运行时库文件和 MFC 文件，

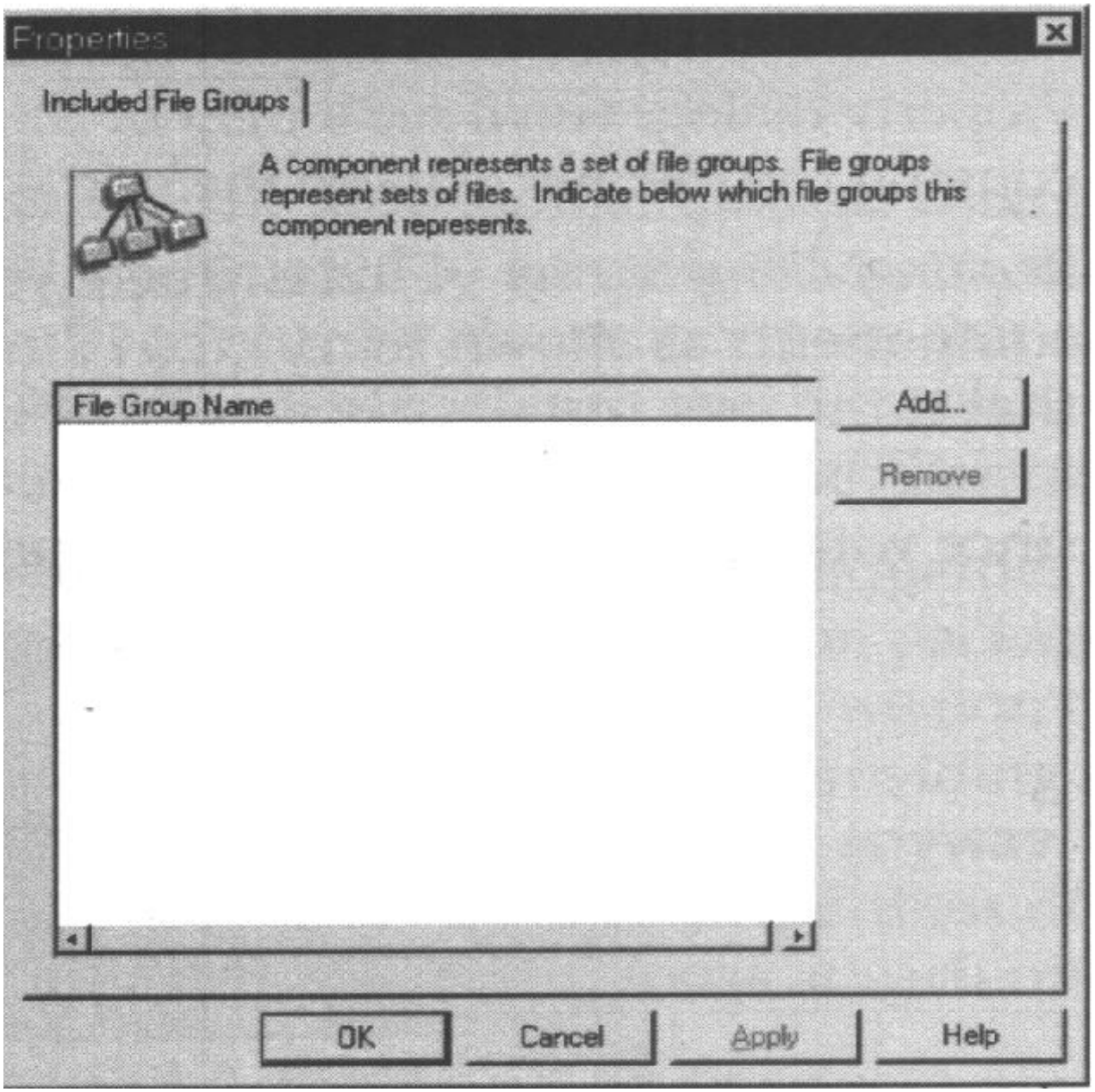
如果每个应用程序都把它们打包后安装到自己的目录下，硬盘空间该是多大的浪费啊？因此，总是把这些共享的 DLLs 放在 Windows SYSTEM 目录下，这时就要更改这个属性（打开属性对话框时，会看到可供拷贝的所有映射地址，选中需要的那个）。

必需的组件　在这里设置组件间的关系，如下图所示。图中我告诉 InstallShield，当用户安装 Program Files 组件时，他或她也必须安装 Shared DLLs 组件（只有在没有 Shared DLLs 组件程序就不能正常运行的情况下定义组件间的关系才有意义）。

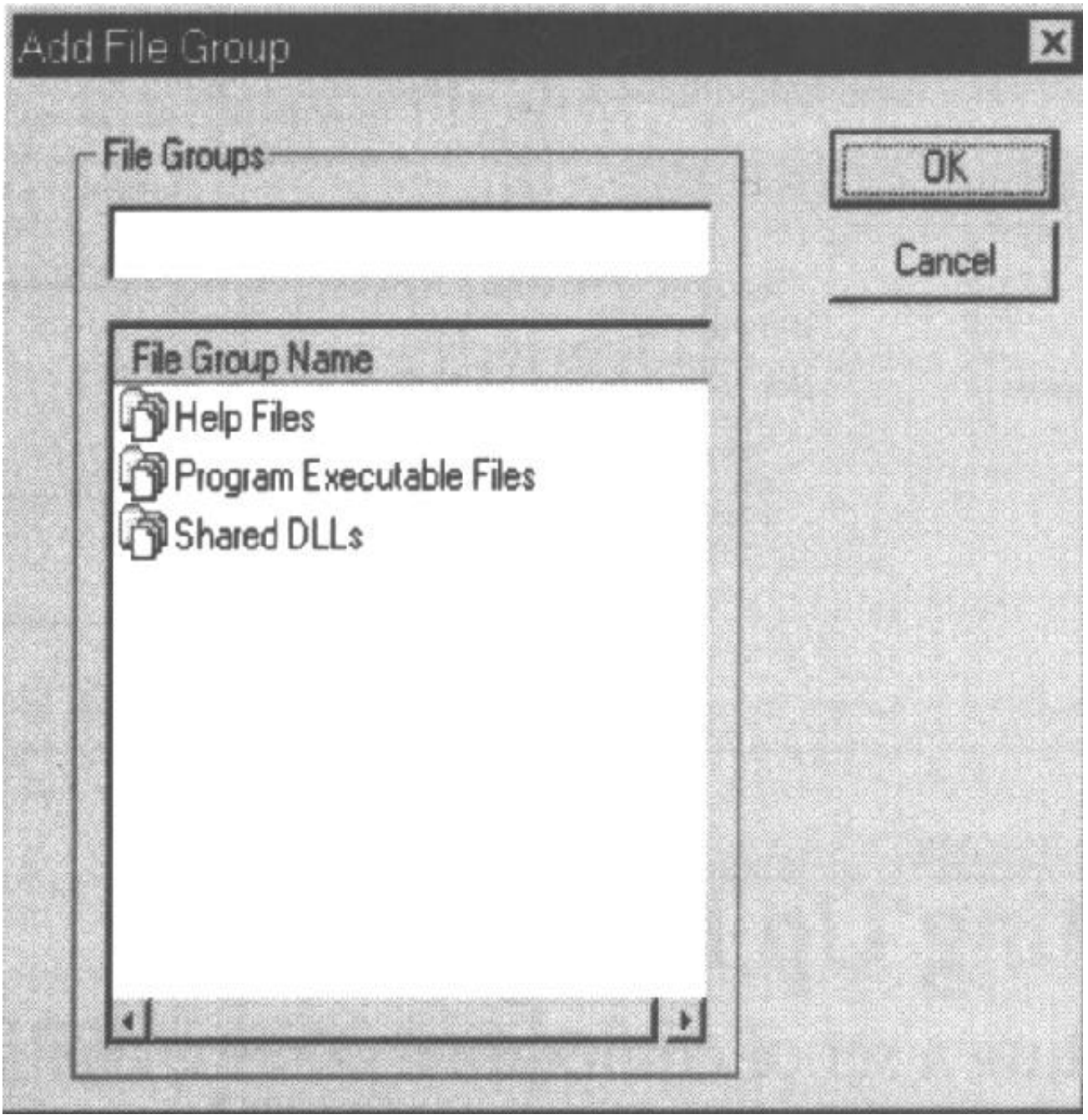


技巧 如果你打算在 Internet 或 Intranet 上安装应用程序，那么请特别留意 FTP Location 和 HTTP Location 属性。这些属性使用户可以从 Web 链接开始安装，并从 Web 服务器上拷贝所需的文件。如果安装文件处于 FTP 服务器的控制之下，那么一般选用 FTP Location 属性。

配置组件时最后一个绝对必须留意的属性是 Include File Groups **属性，直到现在我们还没有给** Program Files **（或其它）组件指定任何东西。双击该属性，你会看到如下图所示的** Properties **对话框：**



Properties 对话框包含了指定给某个组件的所有文件组的清单，前面我们讲过一个例子，安装时可以选择用拼写检查还是用语法检查，它们的 DLL 文件相同，规则文件不同。前面我还讲到过要设置三个文件组：一个带有语法检查规则文件，一个带有拼写检查规则文件，第三个带有通用的 DLL 文件。现在就是设定其间关系的时候了，单击 Add 按钮，你会看到如下图所示的 Add File Group 对话框：



正如你所看到的，Add File Group **对话框包含了我们使用** Project Wizard **定义的三个文件组。在继续阅读本章之前，你需要按下表定义组件与文件组之间**

的关系。表 16.1 给出了本例所用的设置：

表 16.1　Sngl_Doc 安装程序的组件设置

组件	属性	取值
Program Files	Description	All the files required to run the application.
	Status Text	Copying Program Files ...
	Overwrite	SAMEORNEWERVERSION/ SAMEORNEWERDATE
	Required Components	Shared DLLs
	Included File Groups	Program Executable Files
Help Files	Description	Files that show you how to use the program.
	Status Text	Copying Help Files...
	Overwrite	SAMEORNEWERDATE
	Included File Groups	Help Files
Shared DLLs	Description	Common files used by the program.
	Status Text	Copying Shared DLLs...
	Overwrite	SAMEORNEWERVERSION/ SAMEORNEWERDATE
	Destination	<WINSYSDIR>
	Included File Groups	Shared DLLs

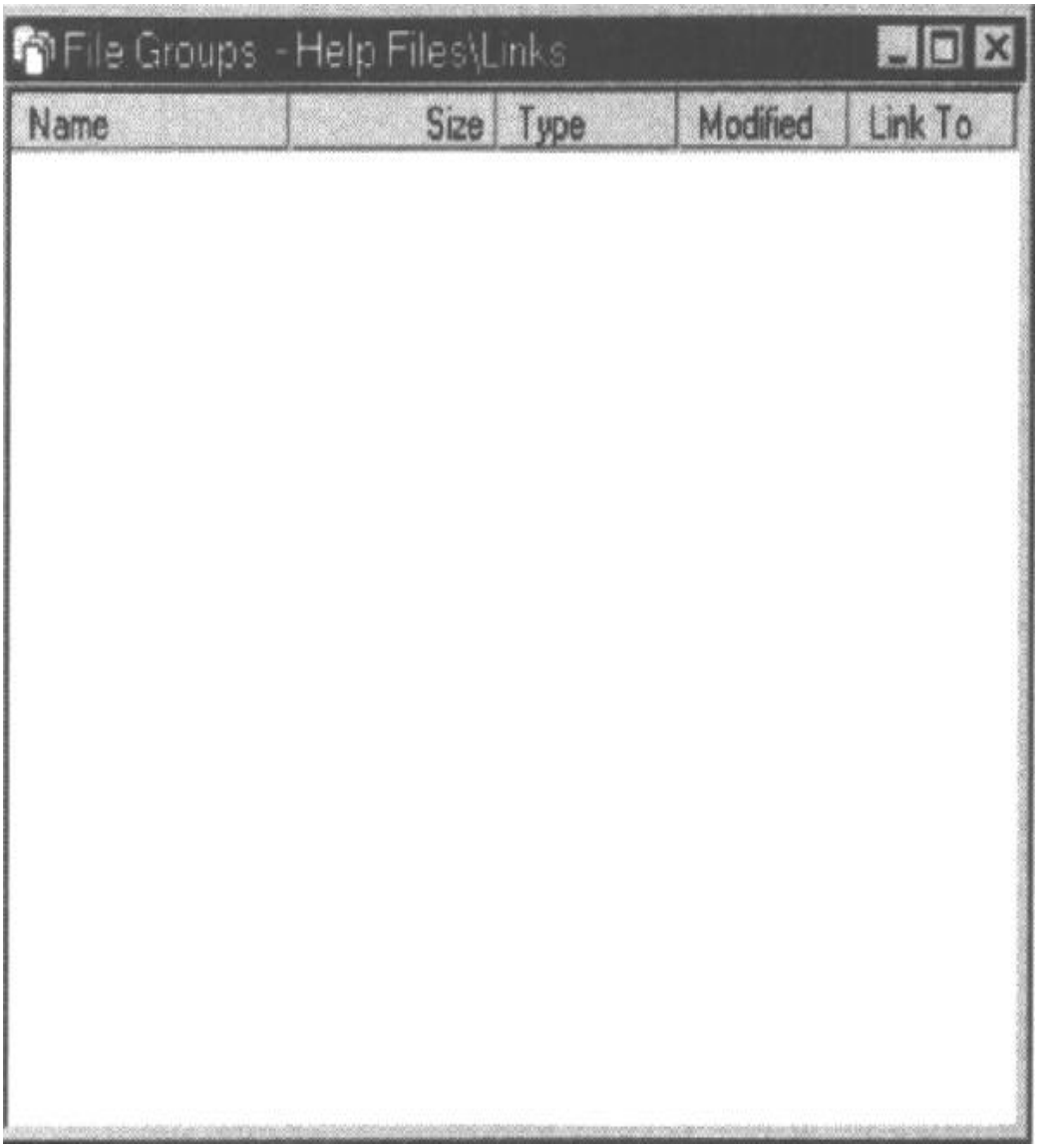
设置文件组

建立一系列的组件对安装程序并没有多大的帮助，它只是告诉用户哪些组件可以被复制到硬盘上，但还没有告诉安装程序要复制哪些文件，现在我们就来做这件事——为前面定义的文件组定义一系列文件。

先单击 File Groups 选项卡，你会看到带有所有文件组清单的 File Groups 对话框，如图 16.3 所示。选择要配置的文件组很容易，单击窗口左边给出的文件组列表项。

单击 Help Files 文件夹旁边的加号（+）标记，再单击 Links 列表项，你会看到如下图所示的 File Groups-Help Files\Links 对话框，从这个对话框中定义该文件组包含的一个或多个文件。

右击 File Groups-Help Files\Links 对话框，从上下文相关菜单中选择 Insert Files，你会看到标准的“打开文件”对话框，找到第 15 章建立的 Sngl_Doc.HLP 文件，单击 OK，就将它加入了 Help Files 文件组中。还需要把 Sngl_Doc.EXE 文件加入到 Program Excutable Files 文件组，最后一个文件组是 Shared DLLs，需要加入下列文件：MFC42.DLL 和 MSVCRT.DLL，它们将放在 Windows SYSTEM 文件夹下。


File Groups - Help Files\Links
Name
Size
Type
Modified
Link To

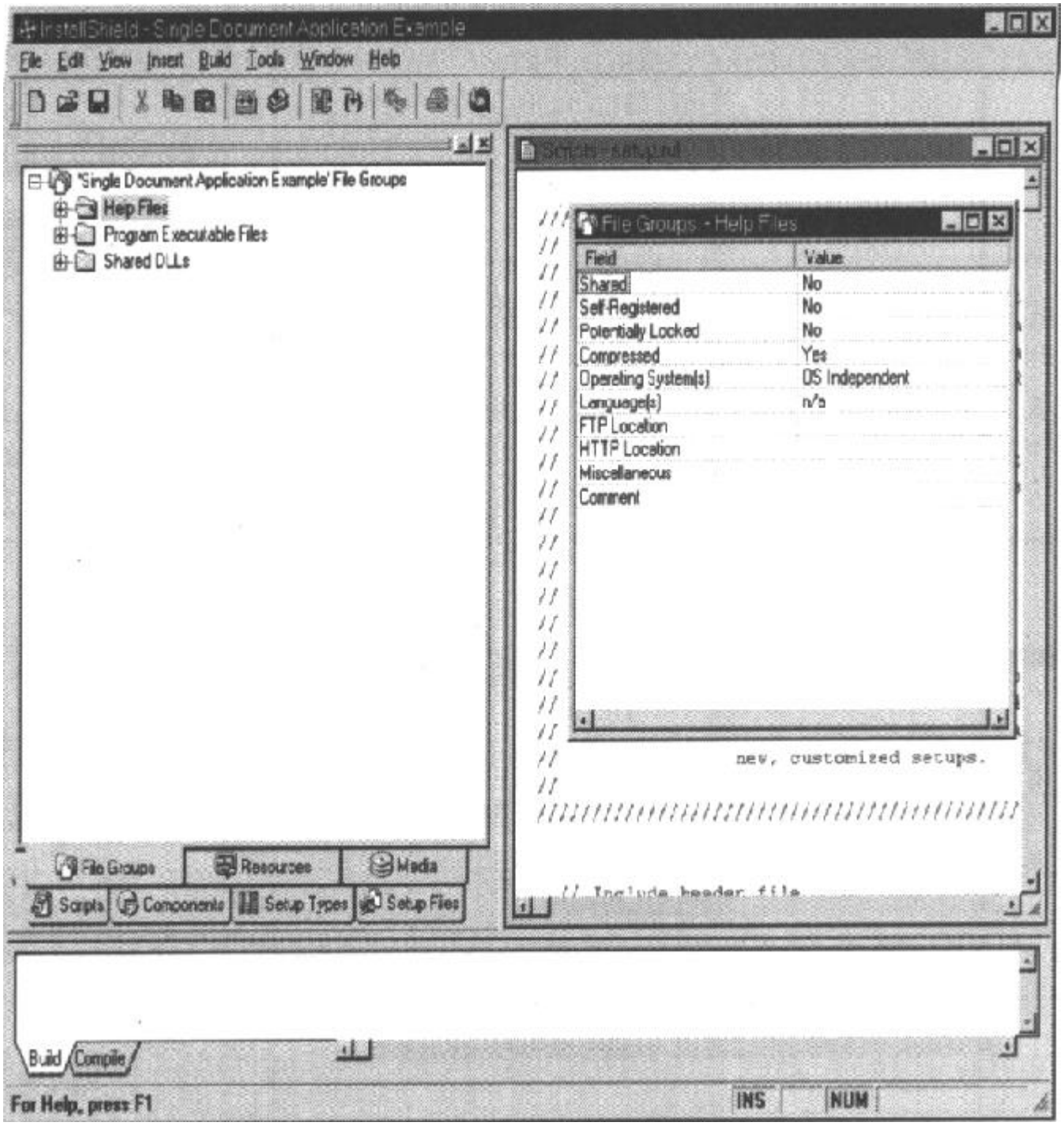


图 16.3　**使用** File Groups - Help Files **对话框配置前面定义过的每个文件组**

定义资源

到此为止，我们已经设置了组件和文件组。下面要给用户提供一些标识信息，因为他们当然希望对所安装的程序有所了解。选取 Resources 选项卡，你会看到如图 16.4 所示的 Resources 窗口，其中含有两个对话框：Resources - String Table 和 Resources - String Table\English。我们要在 Resources - String Table\English 对话框中进行修改。

在 Resources - String Table\English 对话框中进行的绝大多数修改并不影响安装程序，修改的目的是为了帮助用户理解安装程序进行的工作。改变资源值的做法和其它对话框改变属性的做法基本相同，只是双击要改变的 Identifier（标识符），InstallShield 就会显示相应的对话框供修改，下图是修改 COMPANY_NAME 标识符的示例：

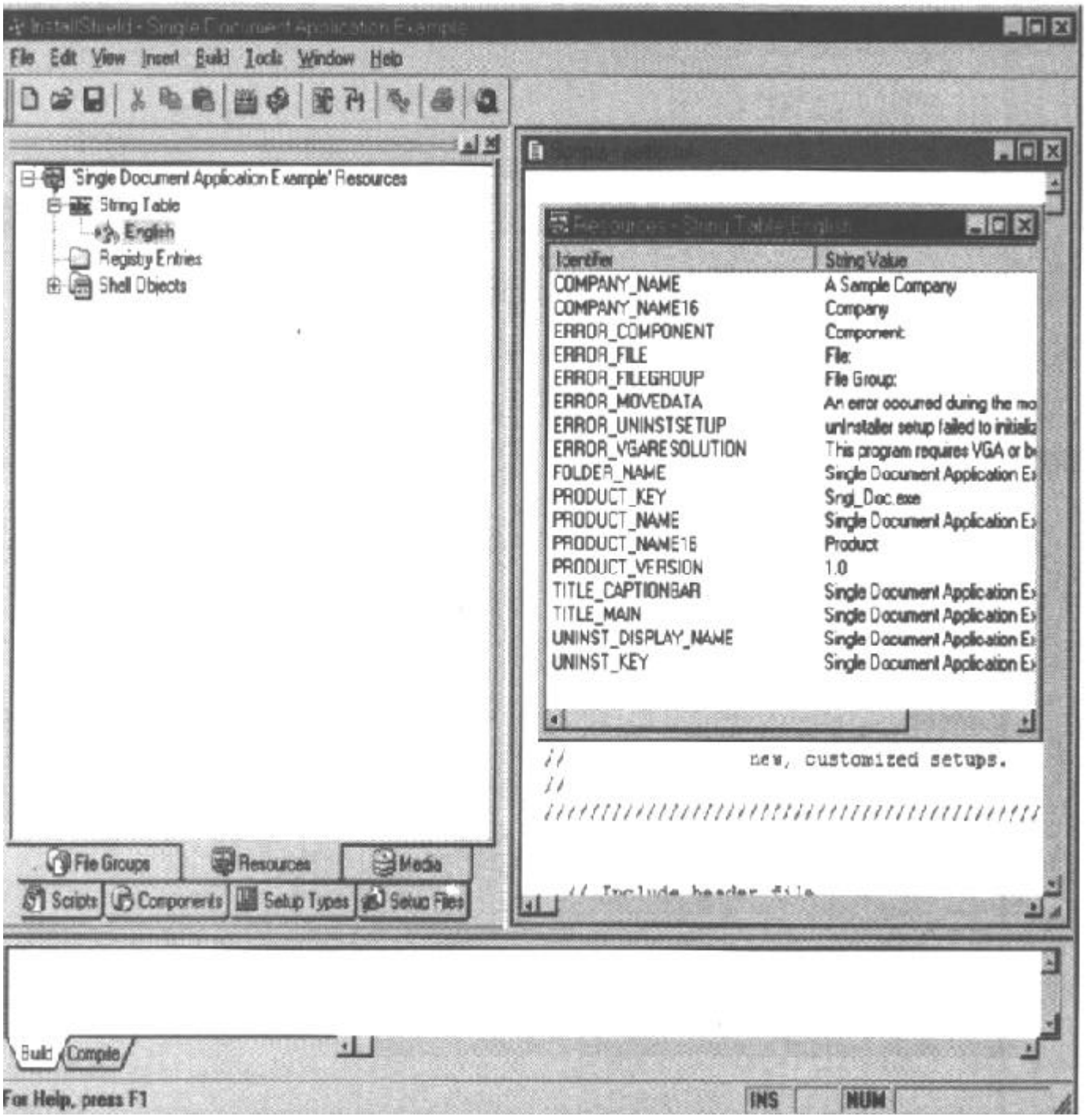


图 16.4　InstallShield **几乎与** Visual C++**相同的方式使用资源**

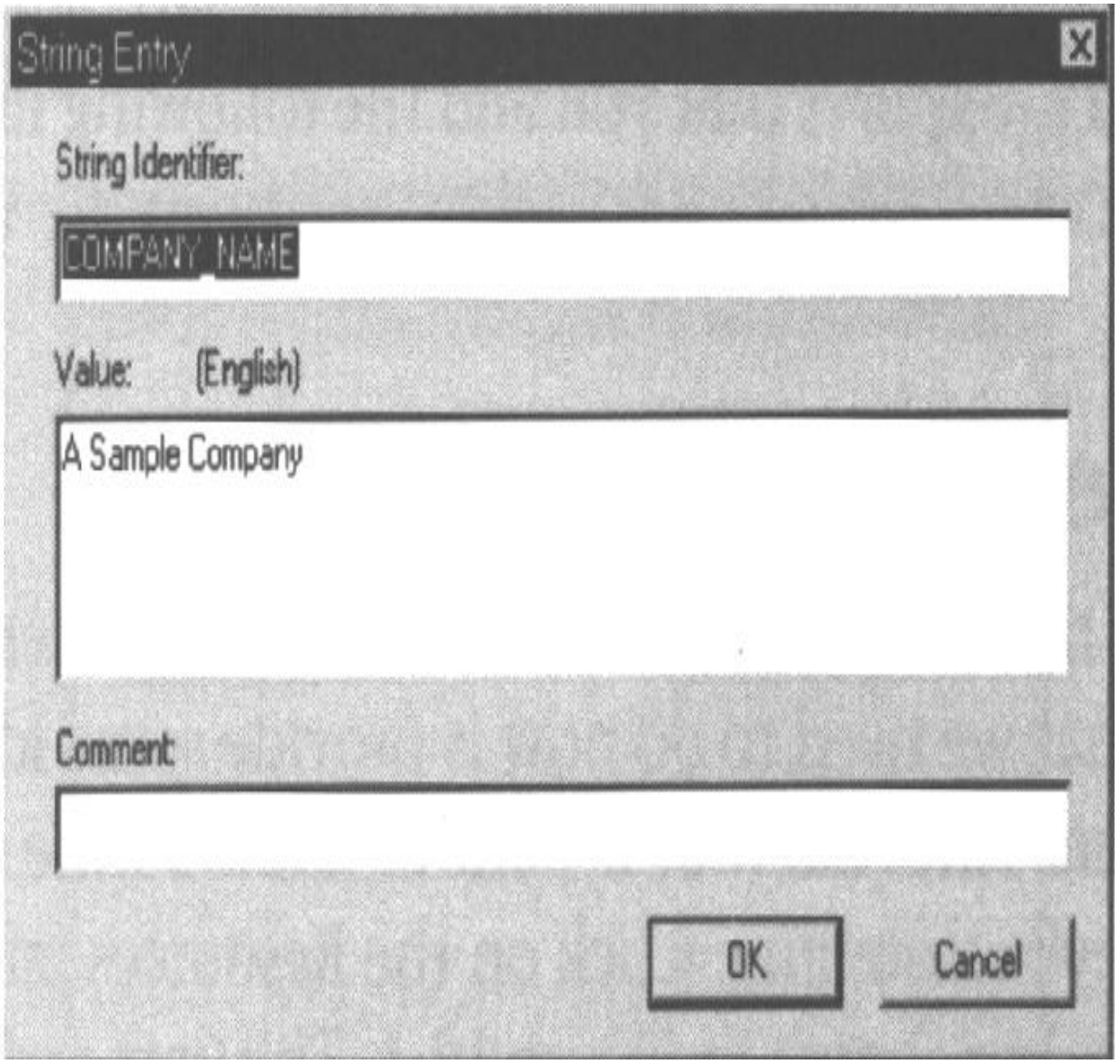


表 16.2 是我们对示例程序所做的修改——显然，各标识符的取值依据安装程序的不同而不同。

表 16.2　Sngl_Doc 安装程序的字符串值

标识符	取值
COMPANY_NAME	A Sample Company
PRODUCT_KEY	Sngl_Doc.EXE
PRODUCT_NAME	Single Document Application Example
UNINST_DISPLAY_NAME	Single Document Application Example
UNINST_KEY	Sngl_Doc.EXE

决定安装介质类型

现在我们要做配置安装程序必须完成的最后一件事了：就是告诉 InstallShield **要使用何种介质发行你的应用程序。在绝大多数情况下，你会使用** CD **作为主要的发行介质，如果应用程序很小的话，也可以使用软盘作为另一种发行方法。如果你只是选择** CD **作为唯一的发行介质，那么现在你不用再做什么了。否则的话，选择** Media **选项卡，你会看到如图** 16.5 **所示的** Media **对话框及窗口：**

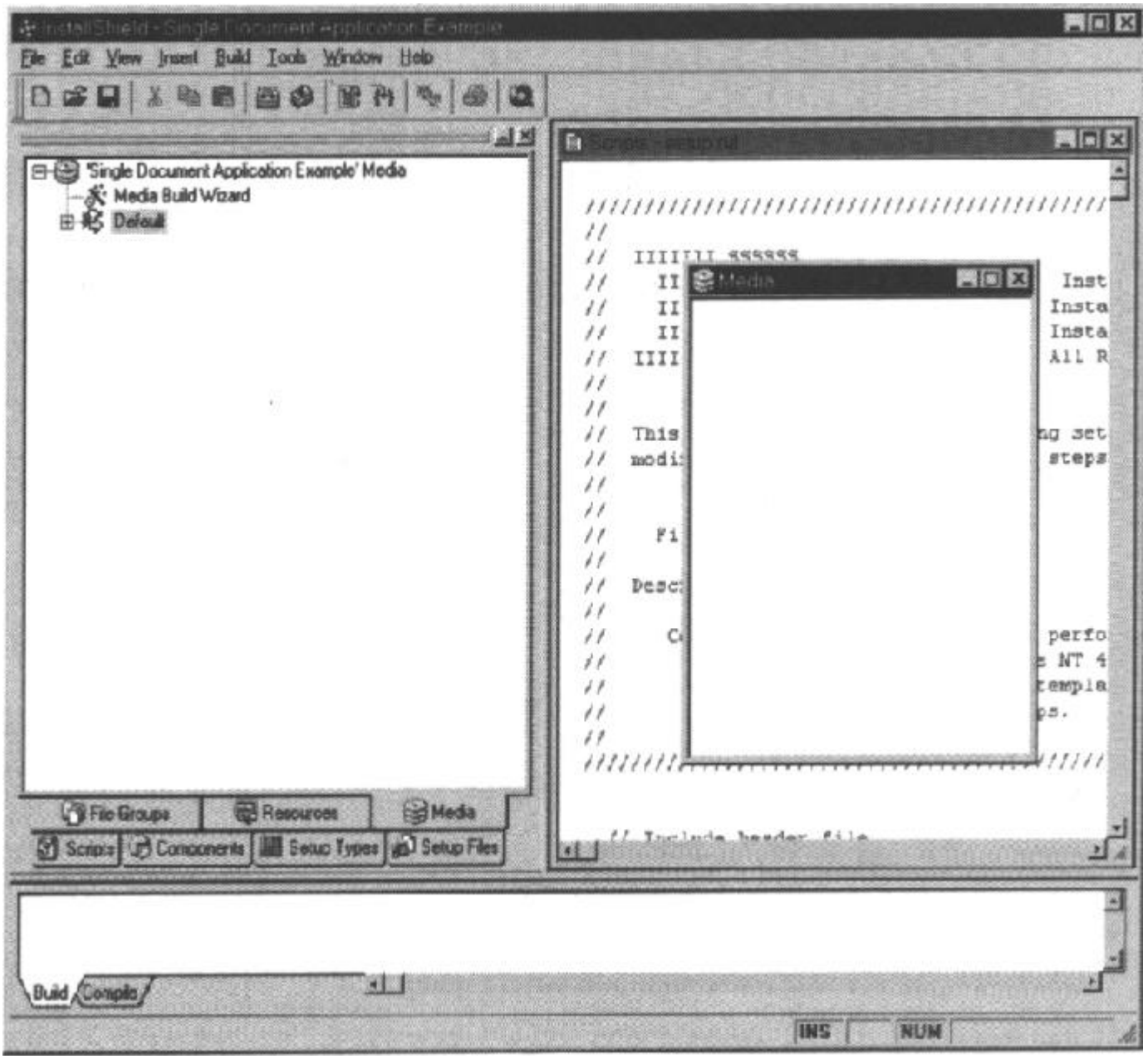
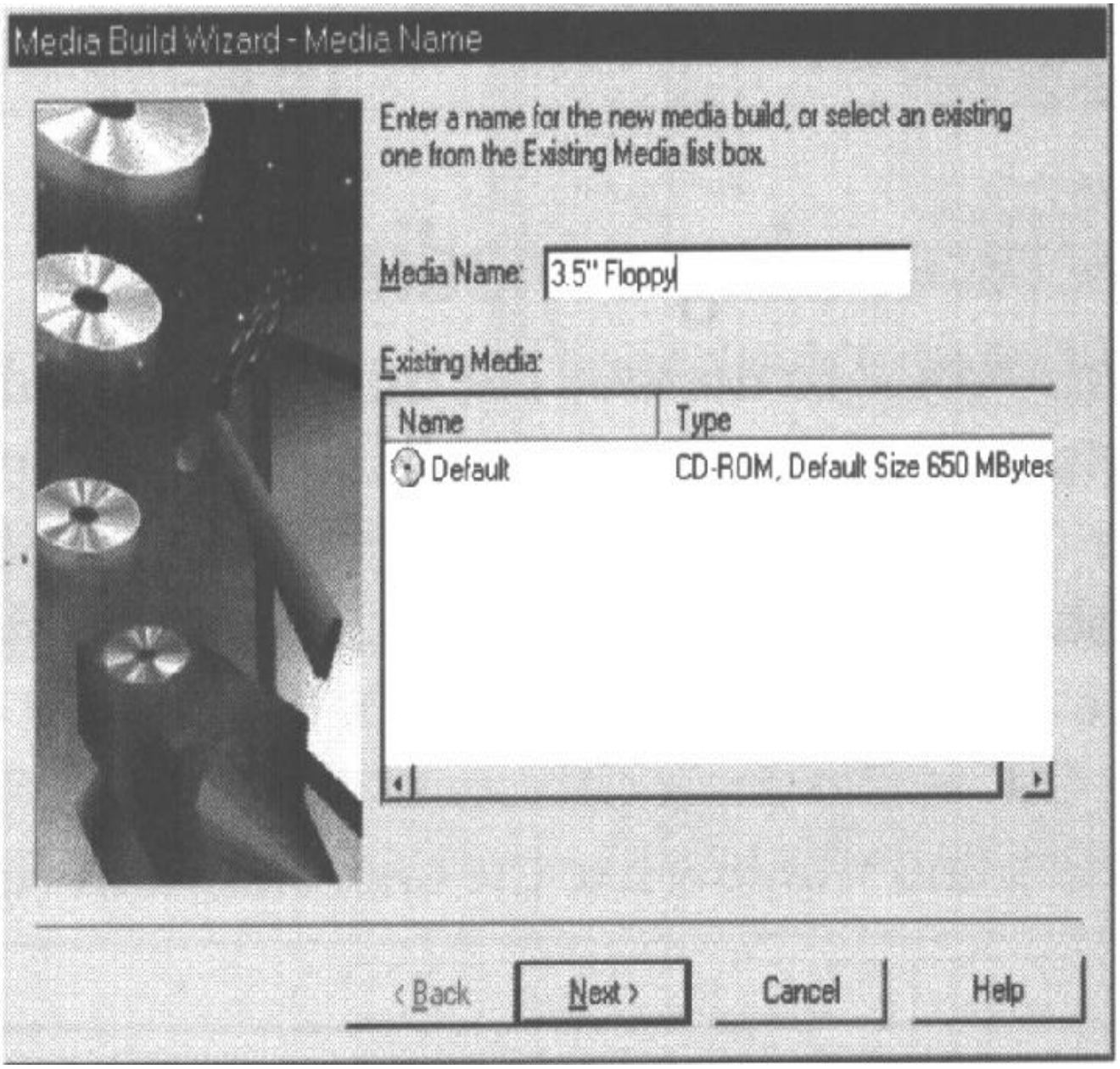


图 16.5　**用** Media **选项卡可以定义除** CD **以外的发布介质**

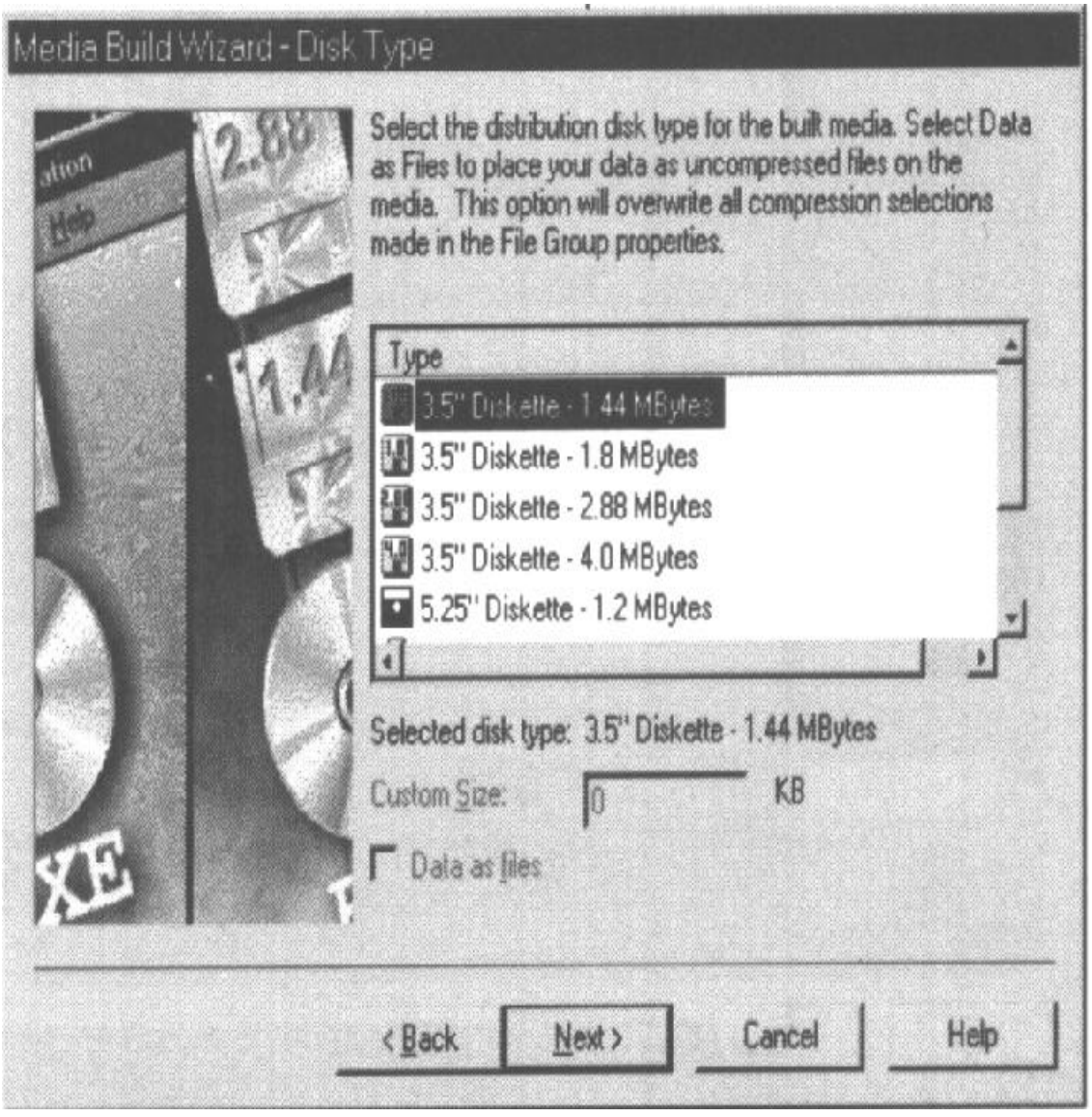
正如你所看到的，CD **已经定义好了（如图** 16.5 **的** CD **文件夹所示）。需要用** Media Build Wizard **向安装程序中加入其它的介质类型，下面的步骤用于**

增加软盘介质类型：

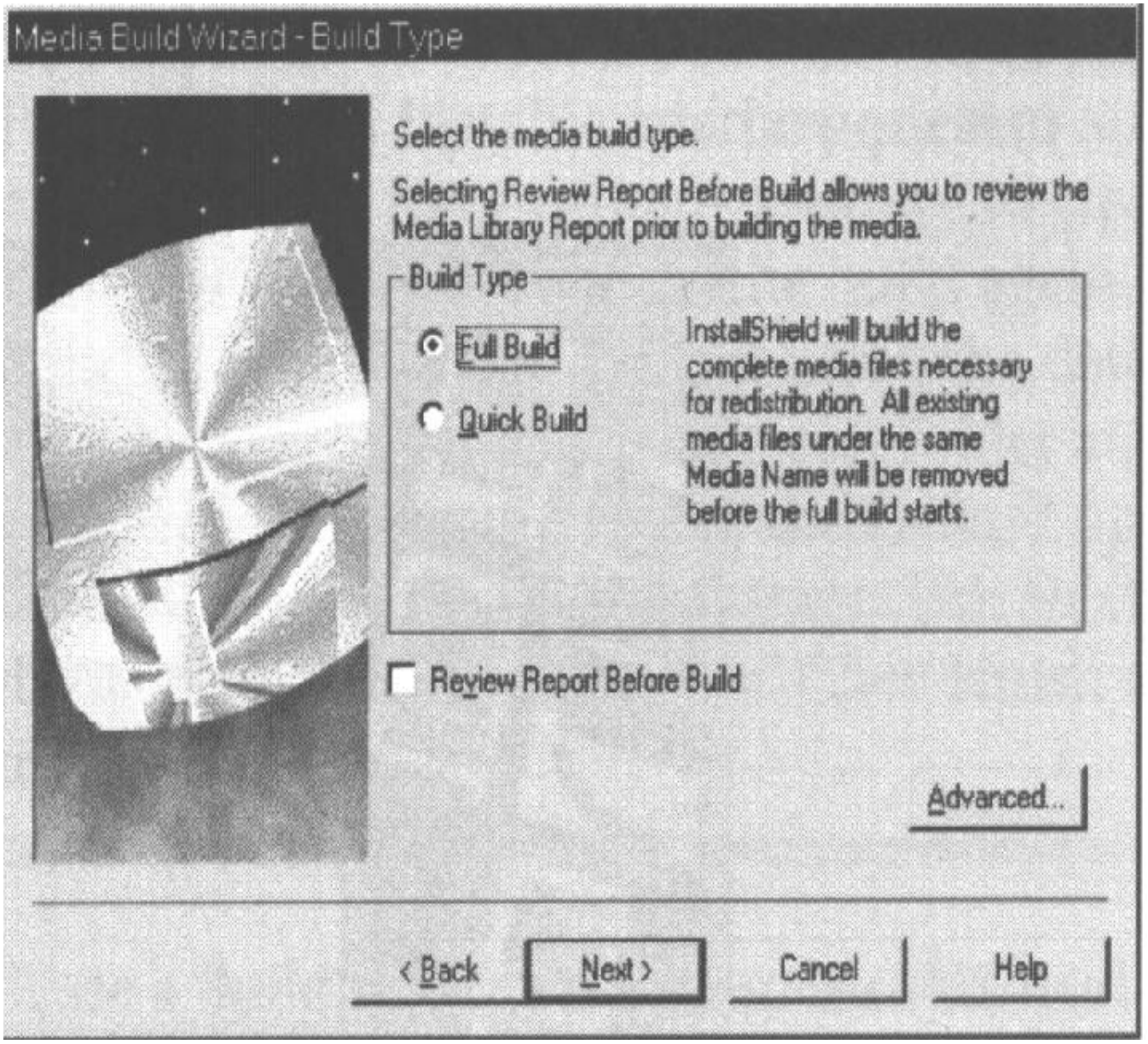
1. **单击** Media Build Wizard **选项，你会看到** Media Build Wizard - Media Name **对话框，如下图所示：**



2. **输入** 3.5" Floppy **后，单击** Next**，你会看到如下图所示的** Media Build Wizard - Disk Type **对话框。请注意，** InstallShield **支持各种各样的介质类型，其中包括** 2.88MB **软盘。**

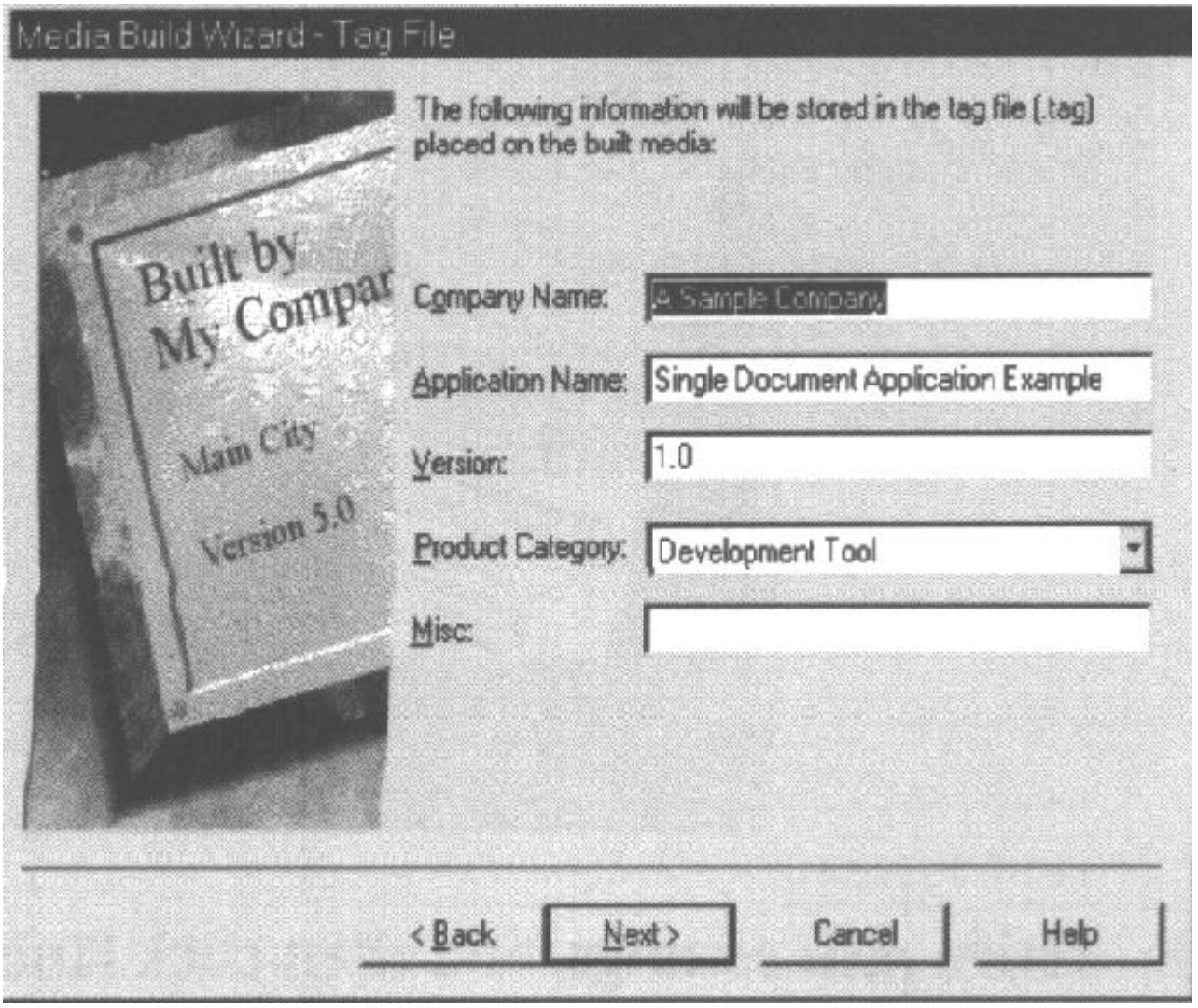


3. **选取** 3.5" Diskette-1.44MByte **选项，然后单击** Next**，你会看到如下图所示的** Media Build Wizard - Build Type **对话框：**



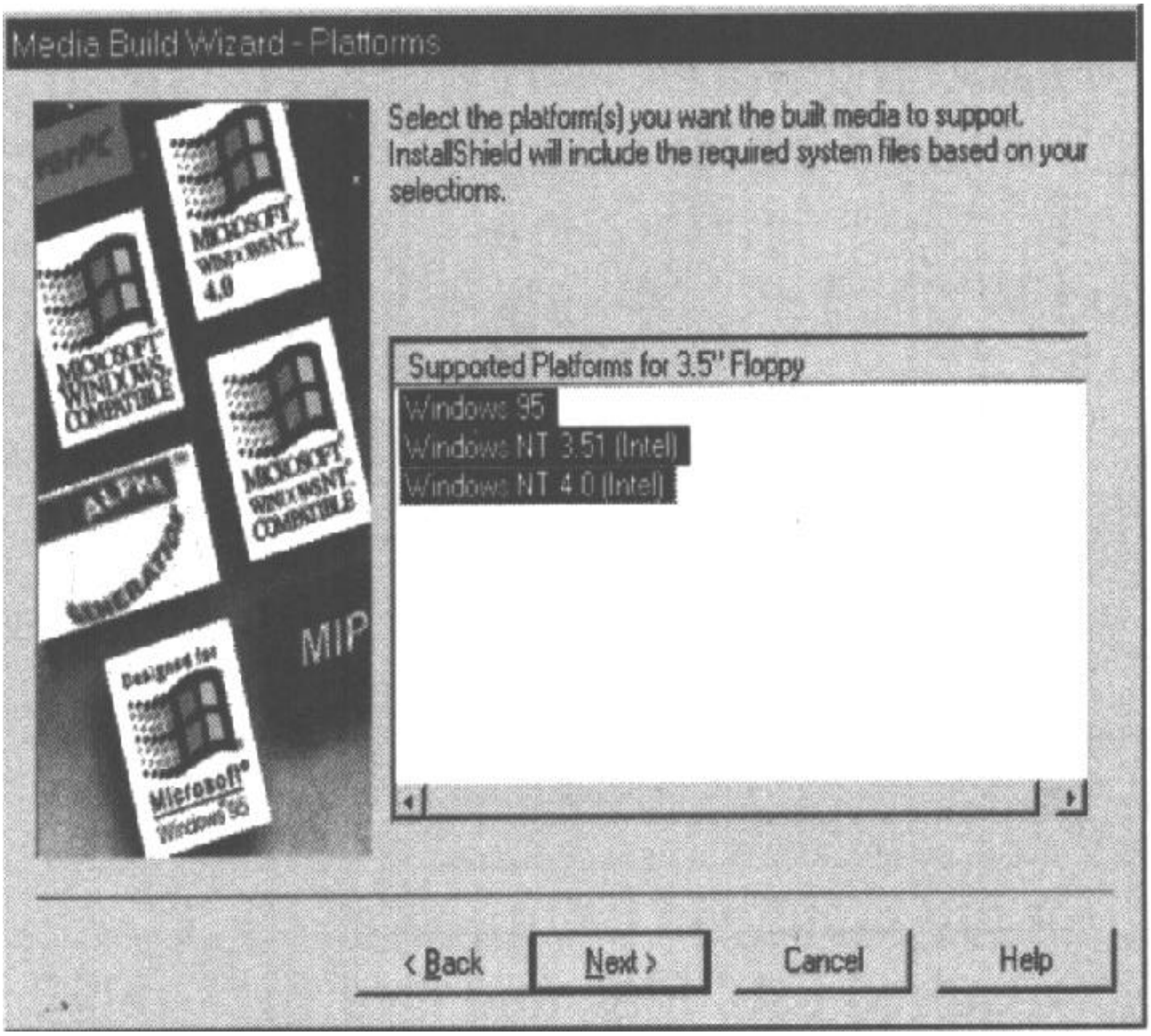
选择 Full Build **选项时，系统将压缩应用程序中包含的所有文件、建立** CAB **文件并创建全功能的安装程序。**Quick Build **选项用于测试安装程序是否正确。**

4. **选择** Full Build **选项，然后单击** Next**，你会看到如下图所示的** Media Build Wizard - Tag File **对话框，从这里输入公司名称及相关的应用程序信息。**

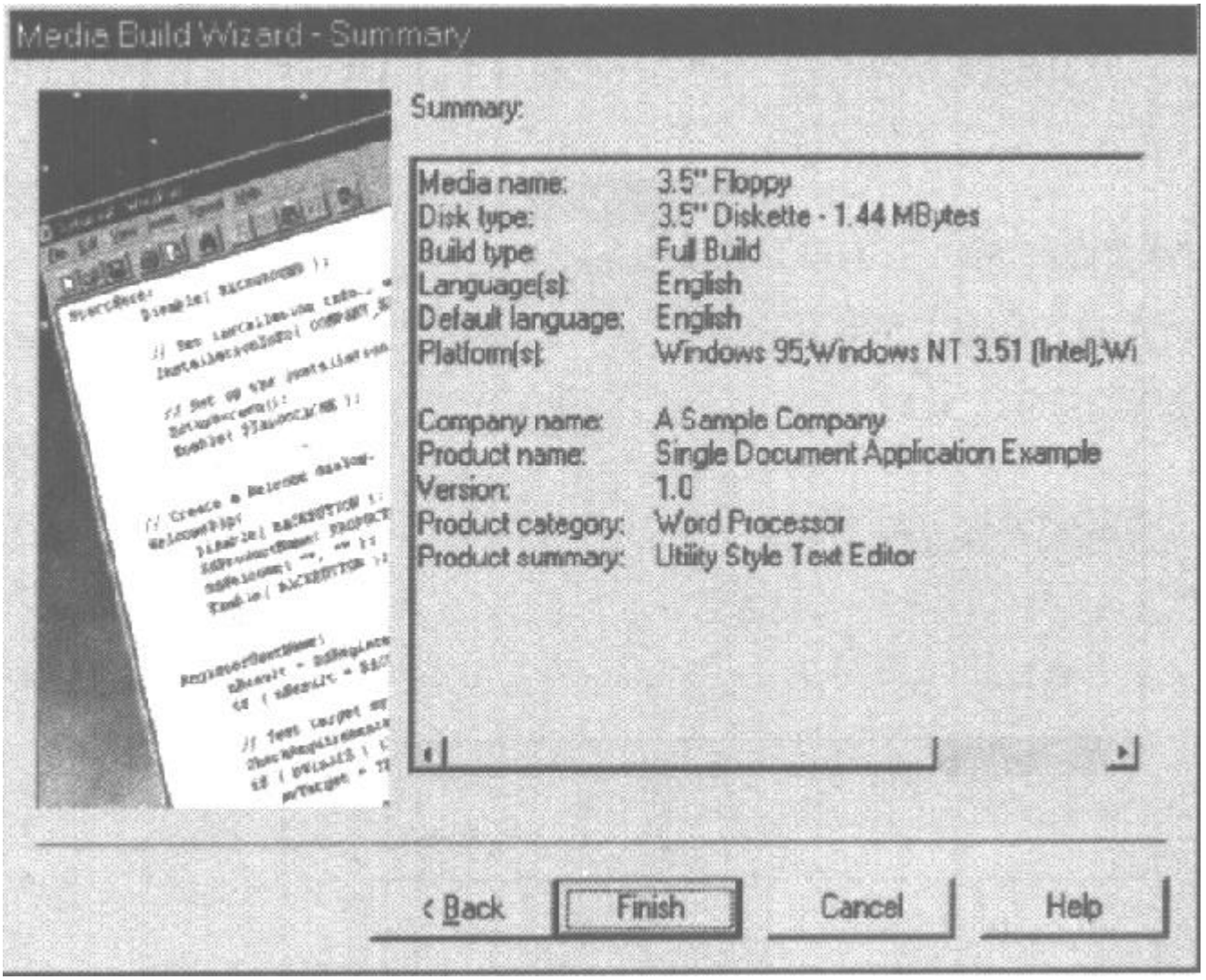


5. **在这个对话框中输入所有需要的信息。示例程序在** Company Name **域使用的是** ABC Corporation，**在** Application Name **域使用的是** Single Document Application Example，**在** Product Category **域使用的是** Word Processor，**在** Misc. **域使用的是** Utility Style Text Editor。

6. **单击** Next，**你会看到如下图所示的** Media Build Wizard - Platforms **对话框：**



7. **单击** Next，**你会看到如下图所示的** Media Build Wizard Summary **对话框，这是验证所有设置正确性的最后机会。**

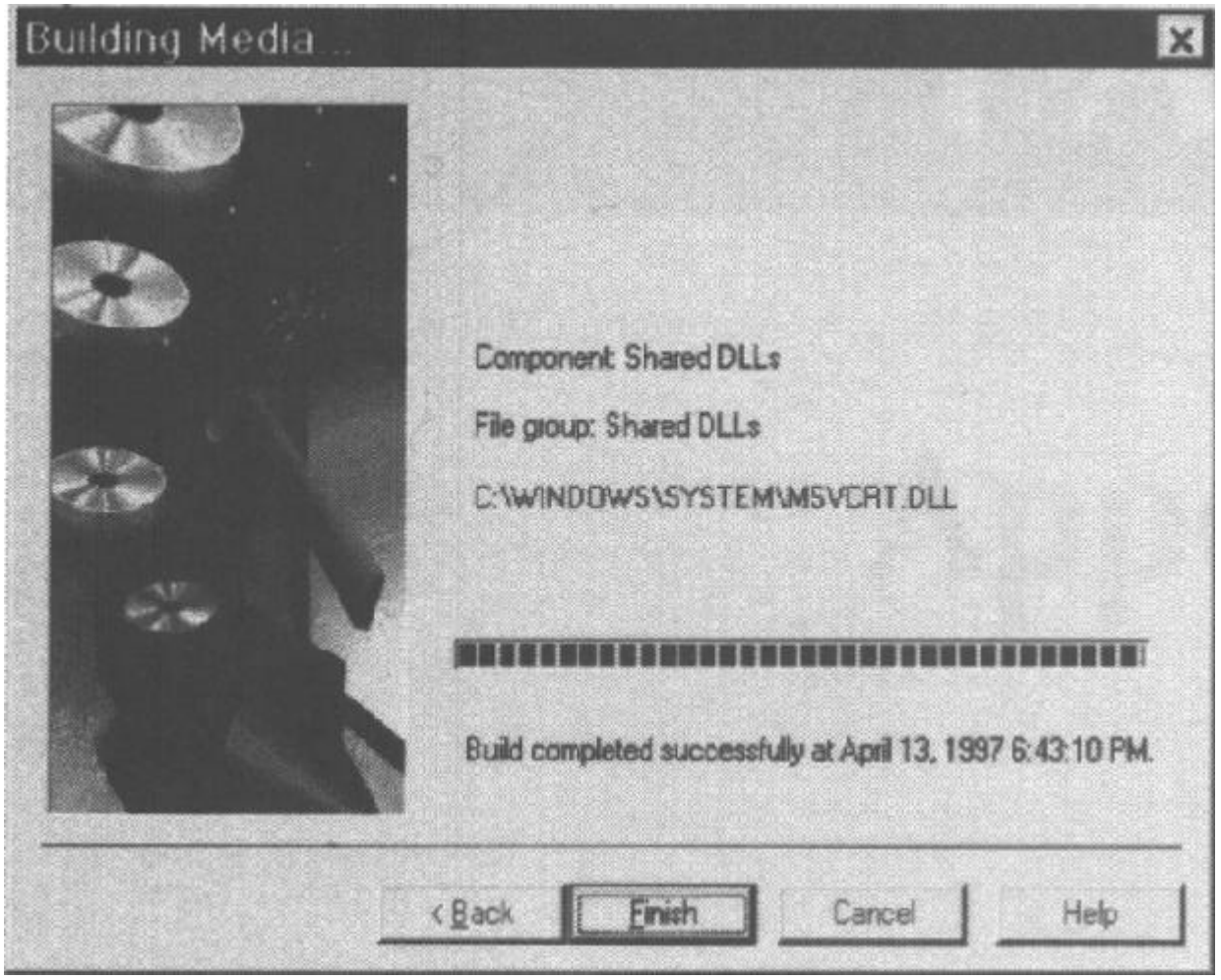


8. **检查所有设置，正确无误后单击** Finish，InstallShield **创建新的、你所需要的安装程序。**

注释 InstallShield 的 Free Edition 版此时会显示一条错误信息，指示没有足够的内存来创建安装程序。可以忽略这个问题，然后重试，如果还不正确，使用缺省的安装设置进行创建。做法是：右击 Default

文件夹，从上下文相关菜单中选择Media Build Wizard，在Media Build Wizard 对话框的第一个显示中选择 Default 选项，然后单击 Next，直到不再出现错误消息，按 Finish 按钮，InstallShield 创建所需介质类型。

这时你会看到 Building Media 对话框，InstallShield 自动创建安装程序，安装程序创建后，Building Media 对话框如下图所示，只需单击 Finish 按钮，这下就全部完工了！



第五部分

附　录



返回总目录

目　　录

附录 A　ActiveX 和 OCX 组件资源 3

A.1　如何以最佳方式购买组件 4

A.2　实用组件介绍 11

A.3　基于 ActiveX 组件的浏览器插件 25

附录 B　在线资源指导 29

B.1　弄清楚你要寻找什么东西 34

B.2　Internet 上的微软公司 45

B.3　微软知识库 52

词　汇　表 56

附录 A　ActiveX 和 OCX 组件资源

用 ActiveX 组件能实现各种不同的目的，事实上，本书第三部分的根本目的就在于此。该部分向你演示了如何在因特网上和应用程序中创建自己的组件，以便完成你能想象到的任何工作。（显然，为了在因特网上应用这些组件，我们还得探索其它一些相关领域，比如脚本（Scripting））。一些人有这样的错误印象，即大多数组件会帮助你把图像管理得更好、提供文件支持、或作为某种实用工具来起作用。这与实际情况相差太远了。你能买到各种各样的第三方组件，其种类之多和你所写的应用程序一样。毫无疑问，你会在某个地方找到一个组件，正好完成你需要它做的那些事情。有时候，你可能会感到惊讶，这些组件实际上是多么的不同啊。

本附录向你介绍如何查找所需要的组件，并且看一看市场上一些有用的第三方组件。尽管许多公司刚刚开始意识到 ActiveX 组件也可以为 C 语言程序员服务，但你仍然可以在你常去的计算机商店或常看的程序员目录的 Visual Basic 部分找到这些产品中的大部分。不管怎么说，本附录是不完全的，它仅仅展示了众多厂家提供的一部分 ActiveX 组件。正因如此， 如未发现所需要的组件，你可随时与下列厂家联系，看看他们能否提供这些组件。

注释　本附录中某些产品来自别的国家，如英国、加拿大。有时候，美国国内的开发人员可能想在本地购买产品，而不是依赖于海外的厂家。

Pinnacle 出版公司也出售这些产品中的许多产品，你可用下面的地址同他们联系（它的 Web 站点是最近才开设的，不可能提供该公司每种产品的信息——如果没有看到你所需要的东西，那么现在就给他们打电话吧）：

Pinnacle Publishing,lnc.
P.O.Box 888
Kent,WA 98035-0888
电话：（800）231—1293
传真：（206）251—5057
因特网：http://www.pinpub.com

A.1 如何以最佳方式购买组件

选择组件可能是项艰巨的任务。在生产组件的厂商所提供的说明几乎完全一样时，你怎样知道这个组件比那个组件要好呢？另外，其它特征，如特许使用费，会使你对某个组件的看法有很大的不同。让我们来看一下在选择组件时应遵循的一些准则。以下部分提供了一份清单，其中包括了你可能想查找的除功能之外的一些东西。这份清单不是无所不包的（有些人可能根据自己的情况有其独特的准则），但你会发现这些考虑会帮助你做出精确的购买决定。重要的一点是，这份清单会帮助你在这样两个组件之间做出选择，其中一个组件提供了许多非常好的特性，而另一个更适合你的需要。

WEB 链接 可在许多地方找到满足需要的 ActiveX 组件；其中有些地方还允许你下载其产品的演示版。你要首先查看的可能是这个站点：http://www.microsoft.com/com/gallery/default.htm，该站点展示了由微软及其合作者提供的 ActiveX 组件。因为微软不包括与 Netscape 导航器一起使用的 NCompass 插件所要求的标记，你需用 IE 来访问该站点。微软站点的优点之一就是厂家通常会预先告诉你控件适用的场合。你可能还想查看站点：http://www.davecentral.com。Dave Central 提供了一个 VRML（虚拟现实模型语言）控件列表，其中包括像通用视频控件这样控件。该站点提供了这些控件的说明，以及到展示其用法的主页的链接和到控件下载站点的链接。位于 http://www.activex3d.com/的 TegoSoft 主页有很多有趣的链接，其中包括一个到三维 ActiveX 组件的链接。

特许使用费

当你把一些公司的产品用作商业目的的时候，他们会因此向你收取使用费。如果你打算在自己的应用开发中用到一个公司的产品，然后发售自己的产品，你可能不得不向该公司支付特许使用费，以获得产品使用许可协议。你一定要首先寻找不用支付特许使用费的产品，如果是那样的话，你就完全不必担心该产品的使用许可问题。如果你在不用支付使用费的产品中没有找到所需要的产品，就应开始寻求 OEM 许可协定。 有时，你可达成缴年费或一次付清的使用

协议。显然，用需要支付特许使用费的产品，将提高自己产品的成本，因此，四处查找不用支付使用费的产品总是个好主意。

源代码

你能购买到的组件中只有少量提供一定级别的源代码支持，要得到其中某些产品的源代码，还需支付额外的费用。如果你是一位专业代理开发商，或是以分售应用产品谋利的商人，与拥有源代码所能获得的利益相比，购买费用是微乎其微的。这不仅是因为拥有源代码，你将学到新的编程技巧，而且你可以用它来增强你所购买的组件的功能。不用说，购买 ActiveX 组件时，源代码是少有的，但确实看到有提供源代码的。通常，应用这些源代码需要具有 C 语言的知识。

对其它程序员的建议

许多人可能都遇到过被充满错误的软件欺骗而上当的情形，这种软件一无所用。不幸的是，生产这种软件的公司通常都擅长于市场营销，他们提供你能想象得到的任何一种东西，而实际交的货却不是那么回事。不久，许多程序员开始想到，在购买产品前，应从别人那里得到一些建议。你也应当如此。可能有人已买了你正打算买的产品，看看你能否在购买之前获得关于该产品的一些意见。

演示版或共享版

有时你可能得不到其它程序员关于某个产品的建议，或许是因为没有人用该产品的方式与你打算用的方式一样。在这种情况下，找到你打算用的产品的演示版或共享版，可减少你日后的许多痛苦。确实，演示版或共享版不能体现该产品的全部特征，而且有些厂家实际上让演示版或共享版将其真正产品的缺陷隐藏起来，但这仍比购买产品而事先一无所知要好。如果你试过了演示版或共享版，看看是否别的程序员同时试过了演示版和最终产品，他们提供的反馈意见将帮助你明白该产品的演示版或共享版与你真正要买的产品之间有多少不同。

技巧 本附录所列的每样产品都包括一个在线服务地址，你会发现大多数厂家将其产品的出版消息及演示版的拷贝放到 Web 站点或论坛。 在购买产品前应当花点钱去访问这些站点。这种访问不仅使你可以找到该产品的演示版，还可以让你知晓你能期望从厂家获得什么样的支持服务。

退款保证

随着软件工业逐步成熟，厂家也变得越来越聪明。如今令人惊讶的是，不少厂家为他们的产品提供 30 到 60 天的退款保证（至少本附录中有个厂家提供长达 90 天的退款保证）。尽管如此，你首先仍需要找到退款保证这一项，确信

在一两个项目中试用该产品后，如果它不像广告上宣传的那样，你可以退货。在通过第三方厂家购买产品时，你也应当了解其退货规定。有时候，厂家并不承诺编程人员通过第三方购买产品的退款保证。因为第三方销售商也没有退款保证，购买者最终只好自己承担购买产品的损失了。

公司稳定性及支持服务

编程人员依赖于厂家通过正规渠道向其提供良好的升级换代来保持竞争能力。如果从一个单人商店购买组件，而该商店明天就倒闭了，你怎么支持用这个组件构建的产品?这个问题不仅延伸到实际事物，如新技术应用等，而且包括当用户向你指出代码错误，而这段错误代码并非出自你的手时，你也需要这种支持。你会发现在你的程序和第三方产品之间的交互作用要比你想象的来得频繁。如果与你合作的公司倒闭了，或者该公司对其提供的支持计时收费，那么等到修改用户报告的错误时，你就不那么走运了。

单个组件的性能价格比与质量

本附录中的某些厂商以非常低的价格提供各种组件的大杂汇。应当明确的是这些组件中有的质量不错，有的则不然。想想你自己的情况吧，你能承受以如此低的价格提供这么多产品吗？毕竟将这些产品归集到一起要花掉不少时间。大多数情况下，你会认为以低价提供这些产品的厂家在某处走了捷径（当然，事物的另一面是有些人要价太高）。如果有人愿意以近乎白送的价格向你

提供某种产品，你在购买之前一定要详细了解该产品。你可能发现，自己只是做了件额外的好买卖，而在更多的时候，你会发现自己购买到了一堆的麻烦。这种组件既不值得你费时费力去安装它，在项目开发中也没有什么用处。

灵活性

图形服务器作为本附录的产品之一，提供了不少于六种级别的平台支持：DLL、VBX、VCL（Dephi 组件）、OCX、FLL、和 C＋＋类（在你读到本文时可能又有图形服务器的 ActiveX 组件问世）。这确实是个额外的好处，因为你永远不知道什么时候需要将为在 Dephi 或 C＋＋中应用而购买的组件用于别的平台，比方说：像 Visual Basic 这样的平台。这对于代理开发商就更是如此，因为他们所代理的工作或客户常常指定了用于该项目的语言。灵活性还表现在其它方面。如果一个通信软件包允许你用到九至十种不同的协议，或者一个网络软件包能与 NetWare3.x 和 4.x 一同工作，你觉得如何?比起购买仅仅满足最低需求的产品，购买那些能做更多工作的产品总是个好主意。

效率

许多组件要求的资源太多，所以你可能不会用它们来构造应用程序，更不会将其添加到 Web 站点。有的厂商试图将编程人员所知晓的每种功能都添加到一个组件中去,这确实能让你只用 10 分钟就写出各种程序,而这种程序需要 16M 内存来运行的事实说明，谁也不能运行你构建的应用程序。另外，当微软开始

写 ActiveX 组件时，它也强调小型模块的价值。显而易见，无论是 OCX 或者别的什么，没有哪种组件需要 16M 内存运行，但仍有很重要的理由来强调这一点。不然的话，你将遇到如下的情形：当你认为自己仅仅给 Web 页面多加了一个振铃或哨声，却发现用户抱怨说程序变慢，在他们的机器上工作不正常。所以，应当寻找短小精悍的组件，用尽可能少的资源完成任务，这时你将发现你的用户很少再有这样的抱怨了。

当涉及因特网时，另一种形式的效率问题表现在为下载而设计控件的方式。你可能会发现，控件自身仅消耗了少量空间或资源，却需要一大堆支持模块的帮助。将一个大组件分解为许多小的碎片并不比一个单独的大组件更好。事实上，如果分解的话，你将发现自己要花费大量时间将所有文件收集在一起，以便到你的 Web 站点的访问者得到完整的控件拷贝。一个高效的 ActiveX 组件通常仅包含一个或两个组件。应当寻找那些花时间生产小规模控件的厂商，如果其生产的控件确实如其所说的一样大小的话。当你得知厂商已将 OCX 组件转换为 ActiveX 组件时，这一点特别重要。要弄清楚这个转换是否仅仅是把名字改变了一下。

注释 本附录中的价格是当前写作本书时所列的价格，如有变化，恕不另行通告。

A.2 实用组件介绍

找到适合你的应用程序的组件可以节省许多编程时间，特别地，如果这个组件是高度可定制的话，就更是如此。本部分包含了一份在现今市场上能买到的流行组件的清单。这份清单不是要替代那些邮寄给你的产品目录，而是让你对现今市场上出售的一些组件有个初步认识。不管怎么说，在庞大的目录中查找组件之前，应当先对该组件有些认识。这份清单也让你熟悉一些很流行的厂商，其中许多厂商提供的组件要比我在此列出的多得多。在试图编写组件之前，你总可以与其中某个厂商联系，看看他们是否已做了你所需要的组件。

技巧 如果你在目录中看到一个控件仅仅以 VBX 的形式存在，花时间与厂商联系一下，你可能会发现这个目录已经过时了，该厂商已经将 16 位的产品移植到 32 位产品之中。即便情况不是如此，你跟厂商联系一下将会使他们下决心做出这个改变。有的厂商是基于（至少是部分基于）购买其产品的用户反馈的意见来做商业决定的。适时的电话联系可能意味着这种区别，即要么你得到所需要的控件，要么自己动手编出这个控件。

注释 本节所列控件，有的刚刚问世，以至于还没有定价。一定要与厂商联系以得到定价信息。你也可以在 ActiveX 组件收藏站点 http://www.microsoft.com/com/gallery/下载其中大部分控件的样

本拷贝。

Graphics Server（图形服务器）

产品：Graphics Server SDK（图形服务器开发软件包）（349 美元）

地址：Bits Per Second,Ltd.

14 Regent Hill,Brighton BN1 3ED,U.K.

电话：01273－727119

传真：01273－731925

CompuServe 网：>MHS:rflowers@bits

因特网：rflowers@bits.mhs.compuserve.com

因特网：http://www.pinpub.com/gserver/home.htm

图形服务器开发软件包是个全功能成熟产品，其运行版本包含在像 Visual Basic 和 Dephi 这样的产品中，该软件包提供了 DLL、VBX、VCL、（Dephi 组件）、OCX、FLL、和 C＋＋类支持，使其成为本附录中最通用的工具。与该产品的运行时版本相似，图形服务器作为独立的应用程序在 Dephi 或 Visual Basic 应用程序之外运行。然而， 该软件包提供了比运行时版本更多的功能。

技巧 对于拥有 Visual Basic 或其它携带图形服务器运行时版本产品的编程人员， Bits Per Second 公司为其提供特别的升级优惠。请与该公司联系以得到当前的优惠价格。

图形服务器是一个 OLE2 基本图形引擎，它支持像画圆或画正方形这样的命令。除了像 Visual Basic 和 Dephi 这样的产品外，你还可以在不直接支持 OLE 的产品中应用图形服务器来完成大量复杂的绘图任务。（CA－Visual Objects 即为一例）。这一特征使得图形服务器不仅仅是一个应用程序编程工具，你还可以在各种不同环境下不同产品中应用图形服务器。

如果你不想与基本图形打交道，而是以图表为目标，那么你可以使用该软件包中包含的 ChartBuilder 产品。该产品是 Visual Basic 中 Graph（图像）控件的全功能版本（图形服务器的最新版本把所有这些功能合并在一个独立的产品中）。只需告诉 ChartBuilder 你想显示的数据位置， 以及在其上面显示何种类型的图形或表格，ChartBuilder 将完成所有的细节工作。

Web 链接 到 http://www.pinpub.com/gserver/gallery/gallery.htm 看一看完整的清单，以及图形服务器支持的图形种类的示例。

要谈到图形服务器的所有特征，仅限于本附录是不够的。下面给出该产品的另一些特征：

- 你可以将图形服务器开发软件包与微软的 Excel 和 Word 一起使用。
- 4.5 版以上产品完全支持 VBA。
- 图形服务器软件包包括对样条图形、浮动条形图，以及所有错误图表上的错误分布图形的支持。
- Bits Per Second 公司计划发行一个 ActiveX 组件，该组件任何人都可以免费下载。一旦安装了这个组件，你就需要在你的 Web 服务器上的图形服务器向该 ActiveX 组件输送数据。

大概只有源代码你不会在此找到。Bits Per Second 想保护自己的投资，所以你只能得到编译过的代码。当前图形服务器提供 30 天的退款保证，并且在将其用于商业目的时，你也不用支付一分钱的使用费。

AllText/Tlist

产品：AllText HT/Pro（350 美元）

TList Enhanced Outline（199 美元）

地址：Bennet-Tec Information Systems

50 Jericho Turnpike

Jericho,NY 11753

电话：（516）997－5596

传真：（516）997－5597

CompuServe **网：**GO BENNET-TEC

因特网：controls@bennet-tec.com

因特网：http://www.bennet-tec.com/

注释 Bennet-Tec 信息系统公司向市场推出了大量的 VBX、OCX 和 Java 小应用程序，本文在此仅提到了其中的两个。一定要访问它的 Web 站点，看看它所提供的所有控件。现在其中大部分控件是 32-位的 OCX，这对于因特网上的应用来说是个不错的选择。如果你需要 VBX 或 16－位 的 OCX，不用担心，该公司对其大部分产品仍然同时提供 16－位

和 32-位的支持。

AllText HT/Pro 本质上是一个较有魅力的字处理器。它包括对超文本的支持、在图形或其它文档中的嵌入式 OLE 对象、RTF 输入输出，以及数据智能支持。PEN 编辑提供了对笔算的支持——如果你要提供这类支持的话，这是个不错的特征。

RFT 是 Windows 帮助编译器所采用的字处理格式，许多字处理器用这种格式作为与其它字处理器交换数据的格式。数据智能支持意味着 AllText 包含将应用程序的名称作为该应用程序创建的某个对象的一部分的功能。理论上，你可以在应用程序中提供一个小型的文本处理器，然后允许用户将文件输出到功能全面的文字处理器而不会发生太多问题，这种字处理器如微软的 Word。

TList 是增强的列提纲控件，它比典型的文字处理器有着更多的特点，这使得它更像在市场上看到的专业的列提纲的工具。其独特之处之一是能定制列在提纲中的项目，例如，你可以给提纲中的某一项添上特殊的颜色，或用不同的图标来显示该项。该产品还支持如下高级特征：项目隐藏、书签、目录映像以及拖放功能。

Communications Library /Fax Plus

产品：Communications Library 3(149 美元)

Fax Plus （249 美元）

地址：Micro Help

4211 JVL Industrial Park Drive ,NE

Marietta,GA 30066

电话：（800）922－3383 **或**（404）516－0899

传真：（404）516-1099

因特网：http://www.microhelp.com

Communications Library 3 **是个通用的通信库，也就是说，该库提供了许多通用的功能，可以构建任何特殊类型的通信程序。该库提供了五种终端仿真：即** ANSI,TTY,VT52,VT100 **和** VT220 **终端。另外，它还支持** 8 **种文件传输协议，其中包括** Kermit **协议、**CompuServe B+**协议、**X modem**、**Y modem **和** Z modem **协议。不像有的软件包，该软件包支持的** Z modem **协议提供了自动恢复功能。使用** Communications Library 3 **进行通讯时最高速度可达** 25.6Kbps**。**

试图支持众多的调制解调器是编写通信程序的一个大问题，Communications Library 3 **提供** 150 **多个初始化串来支持最通用的调制解调器。你还会发现一组** Pascal **子程序和窗体，这些窗体包含了串口及参数选择、电话拨号及许多其它功能所要求的内容。**

MicroHelp **在** Communications Library 3 **中包括了** VBX**、**DLL **以及** 16 **和** 32 **位** OCX**，该库还为所有文件传输提供所需的** OLE 2 **支持。该库还支持多通信端口（实际限制端口个数是** 8 **个）。**

Fax Plus **是** MicroHelp **的另一通信产品，你可以将其应用于任何** 1 **类、**2 **类或** 2.0 **类传真调制解调器。幸运的是，该产品包括你可以用来检测调制解调器类型的组件，如果打算出售你自己的应用程序的话，这个检测是必要的。**

Fax Plus **较为有趣的特点之一是它支持将打印输出用传真发出。换句话说，**

可以按这种方式构建一个应用程序，该应用程序可以接收其它应用程序的打印输出，并将其作为传真发出。也可以移入 BMP、PCX、DCX 或 TIFF 格式的文件并将其以传真方式发送出去。 它采用的标准传真格式是 ASCII 文本格式。

与 Communications Library 3 不同,Fax Plus 自动处理传真调制解调器的初始化。当然，这种初始化局限于它所支持的调制解调器集合，因此在用 Fax Plus 之前，你应当检查一下 Fax Plus 是否支持你打算用的传真调制解调器。该软件包还包含大量的示例程序和其它文档，它们向你演示如何使用该产品。Fax Plus 和 Communications Library 3 不收使用费。

MediaKnife/ImageKnife

产品：MediaKnife(399 美元,OCX 版)

ImageKnife(549 美元,OCX 版)

地址：OLYMPUS Software Europe GmbH

WendenstraBe 14-16

Hamburg,Germany

电话：+49-40-23773-411

传真：+49-40-23773-644

因特网：http://www.olympus-Software.com/

注释 MediaKnife 原先是由 Media Architects 公司推向市场的， 因此你可能将这些产品与该公司联系起来， OLYMPUS 公司作为新的所有者，

与 Media Architects 公司一样，对该产品提供同样的功能和支持水准。OLYMPUS 公司还为拥有该产品老版本的用户提供特别的升级价格——有的价格低到 99 美元。在支付全价购买产品前，一定要检查一下它的 Web 站点，看看有无特别优惠。 你可能还想了解一下它的数字相机控件产品，该产品允许你将数字相机的功能嵌入到你的应用程序当中。

MediaKnife 允许你使用各种各样的文件格式，比如 AVI——来创建丰富多彩的媒体表现形式。你可以使用该产品的基于 WinG 的显示技术来产生图像色彩的过度效果、影像及声音的后台缓冲、不规则的热点区域、以及定制的光标。该产品还提供了动画功能，其中最为出色的是，设计时就可见识到这些功能——这意味着不用花费太多时间来编译应用程序，或者设置一个 Web 站点来测试你的设计效果。

MediaKnife 的特殊功能还包括能够解压缩 Iterated Systems 的与分辨率无关的分形文件，该产品还包括能执行各种任务的编辑器，这些任务包括热点定义，编辑调色板和合成动画。

ImageKnife 是一个非常复杂的图形 OCX 组件，它支持微软的 Access Paintbrush Picture OLE Object 格式，该组件帮助你使用 TWAIN 扫描接口获得图像，这种接口与 CorelDRAW! 和其它类似产品所用的接口完全一样，一旦获得图像，就可以改变其外观，并将其存入文件或者数据库中。运用这些库，就可以得到以 Access 数据库格式存取图形数据库的高级支持，当然，并不仅限于 Access 数据库，还可以将该产品与其它支持这种图形格式的数据库系统一起使

用。

ImageKnife 还能够操作应用程序中常用的图形，该产品支持各种各样的文件格式，其中包括 BMP、DIB、JPEG、GIF、PCX、TIFF 和 Targa。可以将这些文件显示为真彩色（24 位）、SVGA（8 位）、VGA（8 位）、或单色图像形式，这种通用性使得 ImageKnife 成为绝大多数应用中图形库的不错的选择。

MediaKnife 和 ImageKnife 均提供 90 天的退款保证，你还可以将这些产品用在自己的应用程序中而不用支付使用费。

Aditi UI Widgets

产品：Multi Shape Button（多形状按钮），Date Select（日期选择），Masked Edit（掩码编辑）和 SysInfo（系统信息）

（免费下载）

地址：Aditi Inc.

10940 NE 33rd Place,Suite 204

Bellevue,WA 98004

电话：(206)828-9646

传真：(206)828-9587

因特网：http://www.aditi.com/activex/

注释 虽然在写本书时可以免费下载这些控件（这些控件统称作 Aditi UI Widgets 公司的 ActiveX 控件），但并不能保证这些控件总是免费的。

另外， 在因特网的站点上使用这些控件前，应与厂商联系有关使用许可及其它法律方面的事宜。

Multi Shape Button 是能用多种方式调整其形状的按钮，这些方式包括椭圆、三角形、五角形和菱形。如果这些控件形状还不能满足你的需要，你可以自己画出控件形状。与许多其它这类控件一样，Multi Shape Button 允许你给按钮的每种状态应用定制的图标和标题。

Date Select 提供了一个选择日期的选择框。有了 Date Select 控件，就不用手工输入日期，用户只需简单地按加速按钮即可。如果加亮一个单独的日期成分（日，月或年），则当按下加速按钮时，相应的元素会发生改变。单击该控件的另一部分，将显示出一个日历，用户同样可以用它来设置日期。

也许永远无法确知用户在提示符下会输入些什么东西，如果没有一大堆设置错误陷阱的代码，其结果会是灾难性的。Masked Edit 控件将使你从这种繁琐的工作中解脱出来。它严格限制用户的输入，使其与预先定义的输入类型一致。这种屏蔽功能在数据库应用程序中的应用已经有好几年了，很高兴看到因特网上也具有这项功能。Aditi 公司除了允许你根据需要定制输入掩码外，还为你提供了许多预先定义好的输入掩码。

不知你注意到没有，试图为因特网编写应用程序就像在黑暗中射击一样。至少，如果你写了一个应用程序，并且以压缩的形式分发，那么你能够将对系统的要求包括进去。对于 ActiveX 组件来说，情况却不是这样。ActiveX 组件可能会遇到已经问世的任何一种机型，这就是 Sysinfo 控件发挥作用的地方。 该控件允许你扫描客户机，获得其配置的基本参数，比如所用的操作系统以及硬

盘空间的大小等。你还可以检测到处理器类型，以及是否包含有协处理器。一旦检测到这类信息，则当用户因为设备问题而不能使用你的站点时，你可方便地向其提供明确的反馈信息。

SmartHelp

产品：SmartHelp（尚未定价）
地址：Blue Sky Software
7777 Fay Avenue,Suite 201
La Jolla,CA 92037
电话：(800)677-4946 **或**(619)459-6365
传真：(619)459-6366
因特网：http://www.blue-sky.com

注释 本节不准备全面介绍 Blue Sky 软件公司提供的所有工具，这些工具均为面向因特网的。务必访问其 Web 站点，以便了解该公司的其它工具。 可以应用这些工具来提高你的生产效率。

当微软似乎朝着将你从 Windows 帮助文件格式中转移到其它方向时，Blue Sky 公司似乎以同样的决心将你带入微软老的 Windows 帮助文件格式中。SmartHelp 允许你在用户的浏览器中显示 Windows 帮助文件。你可能会问自己，为什么需要这类控件呢？这是因为，目前 HTML 还不提供与 Windows 帮助文件格式具有相同效能的搜索功能。另外，使用用户熟悉的帮助文件格式，确实

能够减少用户寻求支持服务的次数。该控件的独特之处在于它包括了改变按钮位图（该产品提供 48 种位图）以及改变帮助显示类型（Windows 3.x 样式或 Windows 95 样式）的功能。

Light Lib

产品：Light Lib Business（标准版 249 美元，专业版 449 美元）
Light Lib Images（标准版 249 美元，专业版 449 美元）
Light Lib Multimedia（标准版 249 美元，专业版 449 美元）
Light Lib Magic Menus（标准版 99 美元，专业版 139 美元）
Light Lib/400（根据服务器定价，请与厂商联系了解定价细节）
地址：Luxent Software,Inc.
55 Eglinton Avenue E.,Suite 208
Toronto,ON M4P 1G8
电话：(416)487-2660
因特网：http://www.dfl.com

Light Lib 现有五种不同的 ActiveX 组件（其它组件不久将问世），它们是 Light Lib Business、Light Lib Images、Light Lib Multimedia、Light Lib Magic Menus 和 Light Lib/400。该产品的各种版本还出现在编程语言中，例如 CA-Visual Objects 中就包含了这些控件。

Light Lib Business 可以生成图表和统计图。在写本书时，尚不清楚其生成

图表和统计图的确切种数，不过可以肯定，Light Lib Business 包含了基本的 2 维和 3 维图表类型：条形图、线图、饼图、复杂的带图、堆叠图和百分图。当在控件上右击一下鼠标时，其显示的上下文相关菜单也很有特色。该菜单允许用户将当前图像保存为文件或将其作为墙纸放在计算机的桌面上。上下文菜单还包括像 Copy（拷贝）和 Properties（属性）这样的通用项。

Luxent 将其 Light Lib Images 软件作成了一个功能全面的文档及图像管理库。 该软件确实提供了调整图像的各种控件，例如，可以修改一幅图像的亮度及对比度。其初版控件还提供了伽马校正功能。与大多数其它这种类型的库一样，你可以对图像执行种种操作，比如旋转等等。该控件还包括 Light Lib Business 控件中所包括的的特殊的上下文菜单项。

Light Lib Multimedia 软件能使你将动画表现得更好。Luxent 过去常常将声音控件和影象控件作为不同的软件包销售，但后来发现大多数用户最后将两种软件包都买下了。与声音有关的控件提供了一整套典型声音面板的调节功能，这个声音面板看起来很象 Windows 95 提供的音量控制对话框。除了音量控制外，该控件还包括媒体播放设置和文件控制。与影像有关的控件看起来就像 Windows 95 提供的 AVI 媒体播放器， 不同之处在于 Light Lib 的播放器还提供音量控制。该控件还提供速度设置，这可以让你在影像回放时控制其速度。

看着千面一样的灰暗陈旧的 Windows 菜单，你厌烦吗？即使在其中加上图标，还是难以将使用菜单这种工作变得有趣些。Light Lib Magic Menus 控件可以改变这一切。现在，有了 Light Lib Magic Menus 控件，你可以将一幅图像作为菜单的背景，还可以在菜单项上加上真正看起来有趣的位图，而不是依赖于

简单的图标来完成工作。更为最重要的是，对这些新菜单只需编写极少的的程序。

在当今现代化的局域网（LAN）与陈旧的主机之间架起一座桥梁是大多数编程人员关心的问题。Light Lib/400 就是这样一种独特的产品， 它使得架起这座桥梁比以前任何时候都更容易些。该产品多半不会在因特网上应用，这是因为它是为支持客户机/服务器结构的应用程序而设计的。

MicroHelp

产品：各种产品
地址：MicroHelp
4211 JVL Industrial Park Drive,NE
Marietta,GA 30066
电话：(800)922-3383 **或**(404)516-0899
传真：(404)516-1099
因特网：http://www.microhelp.com

MicroHelp 有一个长长的产品清单，比我在本附录中向你介绍的所有产品还多。在最新的一份包含许多不同厂商的产品目录中，MicroHelp 的产品占了 11 页， 而有的厂商的产品连一页都填不满。

MicroHelp 已经完全接受了 ActiveX 技术。 现在就可以访问其 Web 站点，或者到 ActiveX 组件陈列室看看有什么可供使用。你将看到该公司提供了

各种类型的三维控件、一两个闹钟控件、一些滑动杆和屏幕背景，甚至可以看到一些游戏片断，比如纸牌、骰子等。在 ActiveX 组件陈列室中，到目前为止，即使是微软也还赶不上该公司小巧实用的控件产品（这些产品最好称为 Widgets）。

A.3 基于 ActiveX 组件的浏览器插件

插件是在不久的将来使你的浏览器不再平淡无奇的部分原因。你可以在浏览器设置时用插件加进定制的功能。大多数情况下，你会发现插件是免费的，而对插件提供的服务却不便宜。这对于程序员意味着什么呢？这就意味着，如果决定在自己的 Web 站点加进某种特殊的功能，那么你得为其支付服务费。有时候，你可以先付一笔初始服务费，然后每年交一笔维持费用。从用户的观点来看，这意味着他们通常可以免费增强浏览器的功能，获得一个特定的因特网站点提供的附加功能。

本节列举了四种 ActiveX 插件，这些插件在现在或不久的将来会被见到。在因特网站点上使用这些插件时，需要为其提供服务支持，我们总是假定购买这些服务是要花费一定代价的。与所列的厂商地址联系，以得到价格信息。

注释 某些插件取得了如此巨大的成功，以至于它们成了 IE4.0 的一部分或者将被嵌入 Windows 98。

VDOLive/VDOLive Tools/VDOPhone

产品：VDOLive（根据服务器定价——详细情况与厂商联系）

VDOLive Tools（根据服务器定价——详细情况与厂商联系）

VDOPhone（专业版 79 美元，还有其它版本供选择）

地址：VDONet Corp.

4009 Miranda Avenue,Suite 250

Palo Alto,CA 94304

电话：(415)846-7700

传真：(415)846-7900

因特网：http://www.clubvdo.net/clubvdo/Default.asp

VDOLive 的基本目标是让你在自己的浏览器上播放位于 Web 站点上的声音影像资料，它包括播放电影以及其它类型的动画的功能。新的 VDOLive2.0 版保证了更好的影视质量（每秒钟 10-15 帧）和影像大小（352x288）。它还包括一个有趣的想法：故事书模式。在这种模式下，影像仍然与声音同步。实际上，你将看到情节逐渐展开，这一点就像在故事书中看到的那样。

如果打算创建一个庞大的或者复杂的 Web 站点，你会想到购买 VDOLive Tools。VDOLive Tools 包括压缩影视功能以及使 Web 站点界面友好所需的一切。它所用的 AVI 格式与你以前在任何场合曾用过的 AVI 格式一样， 也就是说着你可以将其与其它编程工具一起使用，亦即在任何可能使用 AVI 文件的地方，均可使用该产品。

VDOPhone 被设计为双向会谈产品，它通过浏览器工作，而不是一个单独

使用的产品，这种灵活性意味着可以设计一个 Web 页面，在其中嵌入许多不同的功能， 包括定制的 ActiveX 组件。即便 VDOPhone 不像市场上有的产品那样功能齐全， 但它被设计成可与微软的网络会议软件一起工作。你可以实际召开一个网络会议，在会议期间，如果需要的话，可以提供白板这样的功能。VDOPhone 完全依赖于流技术，这意味着，要应用该产品需要强大的硬件支持以得到优异的图像质量。当对方机器中没有安装影视捕获卡时，也可以将 VDOPhone 以只传送声音的模式使用。

Acrobat

产品：Acrobat ActiveX for Windows 95（免费下载）
地址：Adobe Systems Inc.
1585 Charleston Road
P.O.Box 7900
Mountain View,CA 94039-7900
电话：(415)961-4400
传真：(415)961-3769
因特网：http://www.adobe.com/和
http://www.adobe.com/supportservice/custsupport/LIBRARY/44ae.htm

Adobe 的 Acrobat Reader（阅读器）已经应用于多种场合。如果你是一位程序员，很可能某个时候在自己的机器上已经使用过 Acrobat Reader， 该阅读器以书本的形式阅读帮助文件。Acrobat 使用 PDF（可携带文档格式）格式以显示

设备支持的方式显示信息。 新的 ActiveX 版本允许你用自己的浏览器观看 PDF 文件， 而不是下载文件并且启动整个 Acrobat reader。

附录B　在线资源指导

作为一名程序员，一定知道在需要的时候获得帮助的价值，这比起单枪匹马试图自己解决问题来要有效得多。不管怎样，很可能有人已经遇到过你打算解决的问题，而且正好有你所需要的答案。另一方面，和我一样，大家都知道信息量如此之大，已经不堪重负，想想堆放在桌面上的那摞尚未读过的技术月刊就明白了。在这大量信息的包围中，要找到解决问题的答案，你可能得费力穿越一大堆不相关的信息。换句话说，为了找到问题的答案，你可能不得不一路与多余的信息作战。问题不是你不知道所需要的信息是什么或者去哪儿寻找，而是不知道如何从正确的源头进行跟踪。

注释　在本附录中，我提供了大量的图表和说明，这些图表和说明就像在因特网上看到的那样。然而，在你实际读到本附录的时候，那些 Web 页面很可能已经有所变化，哪怕这种变化只是很少的一点点。因特网是一个流动的资源，因此，这里的大多数图表应当仅仅作为一种例子来参考，与我在写作本书时所见到的这些图表相比，你所见到的内容很可能看起来不尽相同。

除非你隐藏在世界上某个阴暗的角落，否则应当知道 Internet 已经成为大多数人的头号信息资源。我所知道的大部分程序员都至少有一点关于 Internet 的经

验。CompuServe 和其它在线服务也为这些典型的程序员提供了范围很广的信息资源。每个在线服务所提供的信息类型各不相同，但你最终会发现他们多数都提供了某种层次的帮助，有了这些帮助，就可以用来解决我们所遇到的问题了。更重要的是，通过参加讨论近期的工业趋势或者讨论关于编程的技巧，你将会把自己暴露在大量的新思想当中。然而，问题是很多程序员仅仅满足于此，实际上，如果愿意的话，你能够从在线服务或 Internet 上获得更多的东西。下面所列内容会给你一些启发。

接受世界观点而不局限于本地观点　如果你同楼下的朋友谈论关于 Visual C++的编程技巧，你所得到的只是局限于本地的观点；另一方面，如果与像 CompuServe 或 Internet 这样的在线服务上的某个人交谈，你可能得到英国人、澳大利亚人或者日本人关于 Visual C++的编程技巧，这种国际性的交流会使你全面掌握各种各样的技术，对以前可能认为不重要的功能给予更好的重视。另外，实际工作所带来的文化差异会帮助你看到应用这些技术的新方法，这些方法可能是你以前从未曾见过的。

与厂家代表接触　你开始访问 Internet 的原因之一可能是能与厂家的代表进行接触。但多数人对他们真正获得的东西不甚了解。你在在线服务上除了能与其它用户交谈外，至少还能接触到一位厂家代表，有时可能是好几位。有一段时间，我开始与三位厂家代表交谈，有时甚至是四位，当时我正在使用他们公司生产的产品。在在线服务上，你会发现厂家代表所提供的专门知识常常与其它普通用户的观点混淆在一起，其结果是与从厂家技术支持热线那里得到的信息量相比，从在线服务上实际得到

的信息量更大。

厂家电子公告牌（BBS）既活跃效果又好　如果你已经编过一段时间的程序，可能会记得在不久前从厂家电子公告牌那里得到所需要的信息的情形。像以前的厂家电子公告牌一样，在线服务也提供这些功能。例如：你可以上网到 CompuServe、Internet 或 MSN（微软网络）下载任何微软产品的最新的补丁程序。这对其它厂家也是一样的（尽管其中许多厂家仅仅在 CompuServe 或者 Internet 上提供服务。因此你应当查对一下，确保在你得到的在线服务上找到所需要的厂家）。在进入位于 CompuServe 上的 NetWare 论坛时，我常常能找到关于 NetWare 的有用的补丁程序（任何一位编写需要进入 Novell 公司的 NetWare 系统的应用程序的人都知道，能迅速得到所需补丁程序具有重要的意义）。

与硬件厂商接触　有多少次，当你开始使用设备驱动程序或者其它低级代码时，却发现驱动程序不能正常工作，或者需要从驱动器厂商那里获得额外的信息。你如果认为 CompuServe 网或者其它任何在线服务提供的都是关于软件厂商的信息，那就大错特错了。Practical Peripherals、AST、Hewlett-Packard、IBM 和其它硬件厂商也在这里提供很有价值的服务。我认为，每当需要下载一个新的驱动程序或者咨询某个配置问题时，这里展现的关于硬件的信息是非常有用的（如果不曾想到在在线服务上查询与硬件有关的问题，不要以为只有你一个人是这种情况，实际上，有些编程人员对网上关于软件的讨论无所不知，而对业已存在的硬件论坛却所知不多）。与在软件领域发现的一样，在硬件领域，你将发

现同样具有相当水平的专家报告。

注释 不要以为 Internet 是这样一种取之不尽的资源，在那里对所遇到的问题都可以找到业已证实的答案。即使 Internet（或者任何其它在线服务）上的信息是免费的，但是这些信息的准确性并没有得到保证——因为你在使用在线服务时所遇到的人并非个个都是专家，其中大部分都是像你一样的普通用户。然而，在浏览信息时，我曾多次有幸遇上真正不同寻常的人物。事实上，有些人在空闲时间里，经常上网寻找改进其 Visual C++编程技术的方法。在某个地方的论坛上常常可以发现他们，他们在等着与你分享这些信息。幸运的是，你也会在这些新闻组中发现不少专家，他们有些人在此等着寻找好的主意或富有启发的问题。通过向这些专家提问题，你实际上是在给他们提供思想的养料。

让我们进一步说说 Internet 作为广泛的信息资源这个话题。在有些人的心目中，Internet 已经成为九十年代的单身俱乐部，在那里可以遇到新面孔，交换彼此的想法。传媒界没有做任何事情来改变这种观点，他们所做的，只是手忙脚乱地发表其 Internet 站点的名称，这种作法更进一步加强了这种观点。看看你的电视就知所言不虚，在某一个晚上，我和一个朋友看到不少于 80%的新电视节目都给出了一个 Internet 地址，他们将这些作为节目的一部分。有些节目的制作甚至依赖于 Internet 为其提供娱乐信息，Internet 成了这些节目的主要信息资源。当然，电视并不是唯一利用 Internet 的娱乐业，现在，许多游戏及其它产品也将访问 Internet 的能力作为标准的配置，广告及其它几乎任何一种形式的媒体也都

在利用 Internet。

那么，对于典型的程序员来说，充满娱乐信息的站点意味着什么呢？这就意味着，如果不小心访问这些站点的话，你就会遇到另一种形式的信息超载，想想我在第四章中曾提到的 RealAudio Web 站点就明白了。你为下载 ActiveX 控件或插件而将访问的站点中充满了到其它 Web 站点的链接，而这些链接站点与程序员关心的问题毫不相关（很快你就会发现，Internet 不是专门为某一个群体服务的资源，而是每种涉及到的信息的源泉）。简而言之，为了在原始站点得到你想下载的控件，你不得不费力穿越一大堆额外的信息。

同样有趣的一个观点是，Internet 是某种远程通信的魔毯。最近出版界在其出版物中用到“ intranet ”这个词（这个词如果正确使用的话，将会一直使用下去），问题是，如果你看了任何三个人关于该词含义的解释，就会得到三种各不相同的定义。“ intranet ”的真正含义并非为大多数人所想像的那样。我通常将该词保留，作为定义广域网（WAN）的一种扩展形式的名称，它是一种商务工具，允许雇员利用它来分享信息，即使他们在路上也是如此。

当然，Internet 实现了这些想法的一部分，你可以用 Internet 作为极为有用的搜索工具。与别人交换想法总是 Internet 的一部分内容。在远程站点为你的雇员创建一个 Web 站点来检录，这种主意也是可能实现的。然而，这些关于 Internet 的应用没有一种能真正回答你这个问题：即 Internet 都包括些什么东西，以及你如何才能用它来获取最大利益。这些就是本附录所要讨论的，我将花点时间向你介绍 Internet 的基础，以及用来搜索 Internet 的工具。

在读完本附录时，你应当对 Internet 实际所包含的东西有了新的理解。我认

为你将发现这些新的东西要比你曾经知道的多。网上冲浪应当成为这样一种经历，它有助于你实现自己的目标、开阔自己的视野。然而，实际中遇到的问题是，菜单上的选项如此之多，你会很容易迷失方向的。

B.1 弄清楚你要寻找什么东西

如果你像我一样在 Internet 上做大量的研究工作，就会认识到，较快地搞清你所需要的东西是很有益处的。在多数情况下，这意味着找到像 Lycos 这样的搜索引擎，有时候也需要用到像 Deja 新闻（查找包括在新闻组中的信息的地方）这样的专用搜索引擎。首先，当你提出搜索请求时，这些搜索站点给出的结果各不相同。即使这样，这些搜索引擎提供服务所使用的页面结构是一样的，你可以在上面提出自己的搜索请求。其次，来谈一下关键词搜索。执行关键词搜索，你只需在空白处敲入几个关键词，选择一个搜索引擎，然后单击搜索(Search)按钮。IE 或 Navigator 将把你带到合适的 Web 搜索站点，并且开始为你搜寻所需要的信息。

现在就可以尝试一下搜索。选择 Lycos 作为你的搜索引擎（在 http://www.lycos.com 可以找到该搜索引擎），在空白处敲入 ActiveX，然后单击 Search(搜索)（这对任何一个浏览器都适用）。任何一个浏览器在做任何事情之前，会显示一个安全对话框，这依赖于你所选择的浏览器安全级别，可以单击“ Yes ”按钮来清除这个对话框，最后你会见到像图 B.1 所示的 Web 页面。关于这一页面有几件事情你应当注意。首先，你可以将你的搜索细化。也许一

个单独的关键词不足以找到所需要的东西，你必须提供某种更特殊的东西。我发现任何一个 Web 搜索页面都以一定的形式提供了这种功能，事实上，其中许多站点都提供了非常详细的搜索结构。

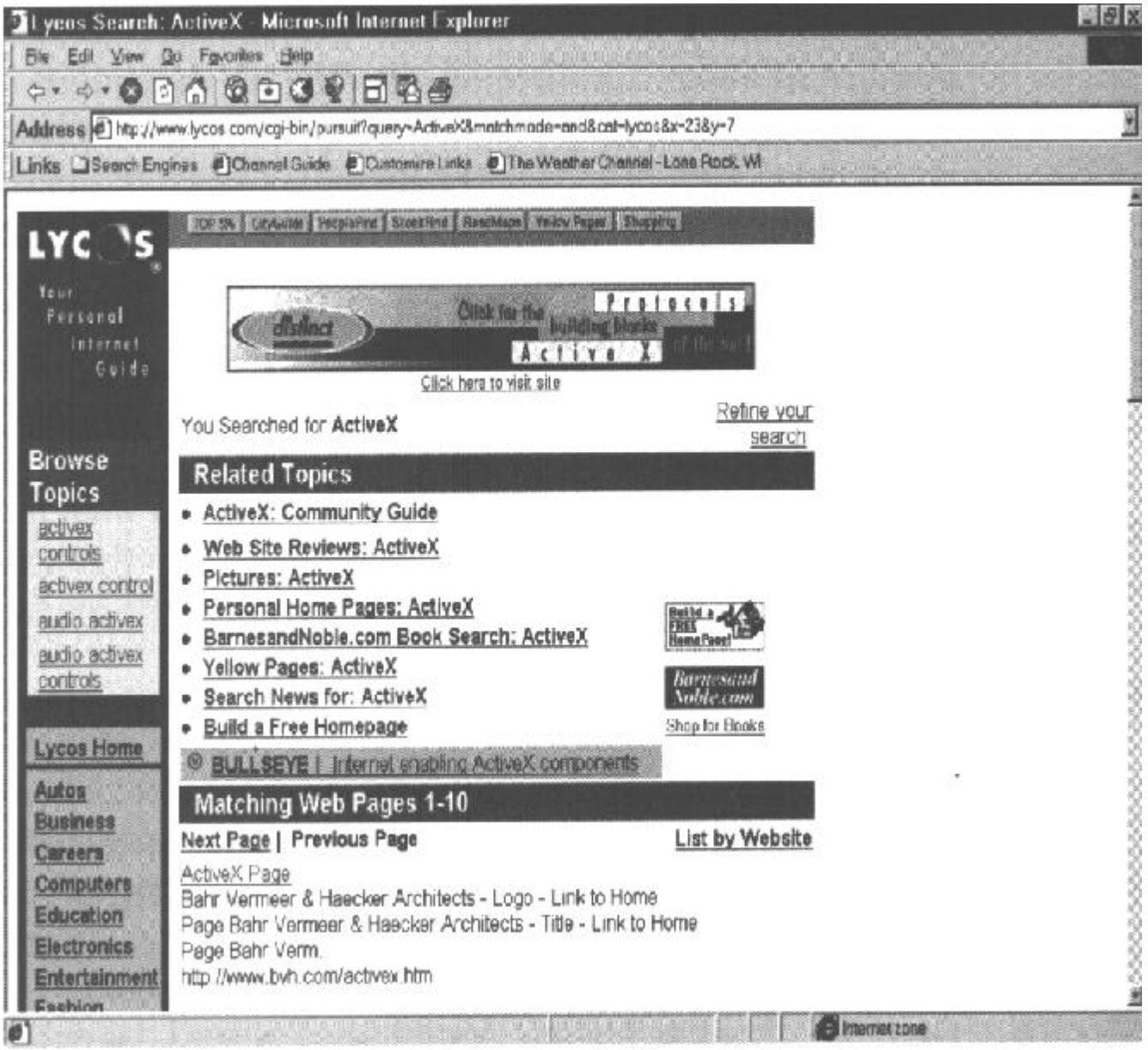


图 B.1　像 Lycos 这样的 Web 搜索页面允许你在 Internet 上找到特定的信息。

还有其它需要注意的事情。每个搜索结果项（也叫“hit”）都包含一个以数字表示的信任等级（有的 Web 站点仅将这个功能作为可选项，因此并非总能见到这个信任等级）。这个信任等级告诉你它所找的结果有多少可信度。通常，搜索引擎应用多种准则来决定信任等级，如关键词在文章或其它信息资源中出现的次数等等。显然，逻辑搜索使得可信度的计算更困难。在你决定应用哪个搜索引擎时，需要考虑的事情之一是该搜索引擎决定搜索结果可信度的因素和方法。我使用各种不同的搜索引擎来达到不同的目的（而不局限于一种）。

像 Lycos 一样，有些搜索引擎还提供特殊的功能，例如，图 B.1 示的 Lycos 页面提供了一个相关主题的列表（这个图中没有显示信任等级）。可以用这个列表快速查找相关信息。不幸的是，这些特殊功能通常工作得不像广告上宣传的那样好。对这些特殊功能提供的信息总是要采取一定的保留态度，因为它没有提供确定其信任度的因素，并且你无法知道该搜索引擎用什么来为你查找这些额外信息作基础。Web 搜索页面通常不将该搜索引擎找到的每个站点都列出来。注意到 Lycos 仅列出其中 10 个，有的服务程序允许你改变所列项个数作为你的部分搜索标准，你必须单击 Next Documents（下面一个文档）按钮来查看在清单上的下一组 10 个站点。你所列项目的不同顺序决定了将哪个站点列在前面，缺省情况下，Lycos 按可信度的级别将其找到的站点列出。这是合乎情理的，因为首先想查看的总是最符合要求的站点。如果你想要字母表序排列，可单击按钮 List Ey Website（Web 站点列表）。有些搜索引擎还提供了其它的排序标准，例如，Deja 新闻站点还允许你按作者排序（我不久要谈到这个站点）。

你可能会想，使用一个搜索引擎就足以满足需要，然而情况并非如此。试

图对这些搜索引擎提出的问题中最容易回答的是，他们每个引擎的工作方式都不相同，一个搜索引擎可能满足我们的需要，工作得很好，却完全不适合于你的工作。因此，我认为重要的是提供一个通用的搜索引擎列表，以及关于他们如何工作的快速索引。我建议你去试试这些搜索引擎，看看哪个能够更好地满足你的需要，以及它们是在什么情况下工作的。下面所列的都是我用过的搜寻引擎，幸运的是，其中大部分引擎均可通过 IE 或 Navigator 的 Internet 搜索按钮来使用。我已为那些想直接访问这些站点的用户提供了网址（URL），只需单击工具栏的打开（OPEN）按钮，输入所示的 URL 来直接进入搜索引擎的页面。

搜索引擎	Internet 地址	说明
AltaVista	http://www.altavista.digital.com/	应用本引擎的好处之一是不会出现信息超载的情况。该引擎返回你所查找的信息项的总量，其服务倾向于以 Web 页面及站点的组合为焦点，这意味着可以用它来得到相当详细的选择。AltaVista 应用摘录来提供信息，这些摘录取自搜索到的文章或其它信息资源。这种服务使用某种秘密的逻辑搜索标准，使得要轻松地缩小你的搜索范围很困难

续表

Deja News	http://www.dejanews.com/	当需要快速查找大量信息时，我使用这个特殊的搜索引擎。当你到达该站点时，会注意到一个高效搜索按钮（Power Search），高效搜索的说法是不准确的。你得自己试一下它能做些什么。除了其灵活性外，这也是一个很容易使用的站点。它使用了大量的图形，其中包括单选按钮和其它熟悉的控件。Deja News 的唯一问题是要获得其提供的每样信息，你得搜索好几次。它提供如此多的搜索选项，你会发现自己时时在考虑用新的方法去搜寻某个特别难以找到的信息。你可获得两个级别的细节，每一样都将告诉你许多内容。获得每一个信息项只能依靠文章题目，这种服务既不提供摘要也不提供综述。它提供的信息确实非常广泛，这意味着你将发现要查找任何一样东西，都将花费时间越过那些确实不适合的信息项

续表

Excite	http://www.excite.com/	本服务倾向于以实际存在的 Web 站点为中心，而不是以关于某个特殊站点的页面为中心。换句语说，你先到达你感兴趣的一般领域，然后 Excite 就可以让你去查找你要找的特殊信息。我发现当我对一个具体的搜索不是很确切地知道时，使用这个搜索引擎是一个优点。在这种情况下，从广义看，它将帮助我看到所有有用的东西，并以此为基础做某些改进。Excite 确实为你要搜索的站点提供概要信息，它倾向于以讨论组和厂商信息为中心

续表

FTP Search v3.3	http://ftpsearch.ntnu.no/ftpSearch	**这是一个非常特别的搜索引擎。该引擎集中于FTP站点,而不是在Web站点或新闻组内查找你所需的信息。可以用它来找一些特殊的文件，例如，你可以用它来找一个有用的东西，你特别需要用这个东西来完成你正在编写的程序。你也可以用它来搜索设备驱动程序、补丁程序、或任何其它你需要的程序**
Infoseek	http://guide-p.infoseek.com/	**这个搜索引擎强有力的地方是只提供事实，它使用能够存取的文章或其它信息资源的摘录。其提供的信息项要比有些搜索引擎详细，这是因为 Infoseek 集中于 Web 页面而不是站点，唯一问题是在这个搜索引擎上缩小搜索标准的能力非常有限**

续表

Lycos	http:www-msn.lycos.com/	我所用过的搜索引擎当中，Lycos 提供的信息似乎是最广泛的，它同时对 Web 站点和页面进行分类（而只要可能的话，集中以页面进行分类）。Lycos 为描述一个特别信息项的内容而同时提供综述和摘录，缩小搜索范围的能力超过现今可用的大多数搜索引擎。应用这个特别搜索引擎的不利的一面是细节太多。如果你对自已寻找的东西没有把握的话，你会很快发现自己在错误的方向搜索，最后得到一堆完全无用的信息

续表

Magellan	http://magllan.mckinley.com：8080/	你会趋向于用这个搜索引擎来查找秘密的信息资源。它提供的信息库似乎不是非常广泛，但它通常提供关于查找的东西的有趣事实。Magellan 集中于 Web 站点而不是页面，因此你得到的站点是非常广泛的，同时它提供的缩小搜索范围的方法比较少。这个搜索引擎依赖于综述而不是摘要来反映信息项的内容。其中一个有趣的功能是其加速搜索一个特殊站点的方法：在 Review 按钮上单击一下，将显示出一整页的综述，该综述给出了其中信息与其它站点相似信息的联系

续表

MSJ Tools for Windows Search Page	http://www.microsoft.com/msj	对于 C++程序员来说，这是一个必须访问的站点，它是由《微软系统月刊（Microsoft Systems Journal）》支持的特殊站点，在该站点可查找任何类型的 Windows 工具。关于该引擎的有趣的事情是你可实际区分不同平台来进行搜索，例如，你可告诉它你只对 Windows 95 工具感兴趣。你也可以指明一个产品类别，其中包括像 ActiveX 控件或其它公共数据开发工具这样的主题。如果你明确想缩小你的搜索范围，该引擎甚至还允许你指定一个公司的名字

续表

Open Text	http://search.opentext.com/	这个独特的搜索引擎是易于理解而且灵活，我发现它是很容易使用的站点之一，而且提供选择的信息库相当广泛。在大多数情况下，Open Text 依赖于极短的摘要来提供信息，它集中于 Web 页面而非站点，这样你将得到相当窄范围的结果
SHAREWARE.com	http://www.shareware.com/	编程人员倾向于依赖大量共享工具来完成一定的任务，如画图标，这个搜索引擎专门帮助你寻找独特的共享软件来完成工作，不幸的是，它似乎更专注于 HTTP 站点而非 FTP 站点(如果你非常需要查看 FTP 站点的话，用 FTP 搜索引擎)

续表

WebCrawler	http://www.webcrawler.com/	你将发现与其它大多数引擎相比，要使用该搜索引擎需要做更多的工作，因为它不以摘要或综述的形式提供信息，另一方面，它提供一个完全逻辑搜索引擎，以及一个相当广泛的信息库。在本列表中唯一比该引擎提供的信息更多一些的搜索引擎是 Lycos
Yahoo	http://www.yahoo.com/	在所有列出的搜索引擎中，Yahoo 提供的信息组织的最好，它以多种方式将信息项分类，使你有很大的机会在该搜索引擎中找到信息。尽管如此，它不像其它搜索引擎那样提供大范围的信息。它还依赖于很短的综述来向你介绍信息项的内容。大多数情况下，我将 Yahoo 作为首先查找的搜索引擎——该引擎将对一个关键词给出广泛的背景材料

B.2 Internet 上的微软公司

如果最近你没怎么听说 Internet，那你一定是生活在没有电视的荒岛上。我时常惊讶于如今 Internet 所面临的压力，并非所有这些压力都来自微软。在很久

以前，微软就开始了一场向 Internet 转移的运动，现在为了侧重于 Internet，它甚至放弃了任何其它形式的支持。如果你确实想了解任何发生在微软的事情，Internet 就是应当去的地方。微软在 Internet 上的主要站点（URL）是 http://www.microsoft.com（其主页实际上在 http://home.microsoft.com）。显然，这个站点仅是来自微软的信息冰山之一角，但却是个好的开端。

技巧 在写作本书时，Internet 是从微软获得补丁程序及信息的最好地方。CompuServe 过去曾是用户与微软联系的地方，但微软已从那里转移到了 Internet 上。老的应用程序的文档仍将 CompuServe 列为一个信息资源，并且它确实是，但你在那里不再会获得厂商明确的帮助。WUGNet 确实包括许多论坛，这些论坛是你过去常常在 CompuServe 发现的由微软雇员设立的，因此在你的手册中，CompuServe 的 GO 命令仍起作用，另外，微软继续在 CompuServe 上保留一定的服务，如微软知识库等。

迈好第一步

要更好地与微软支持的所有站点保持一致是困难的。Internet 的形式是完全自由的，这就是为什么它是提供信息的合适地方。另外，改变你的站点结构的方法取决于你自己的服务商。像 CompuServe 这样的在线服务商为自己保留了这种能力，这使得厂商做任何改变时要多想一想，因为微软控制了其 Web 站点的格式及表现，它可以改变它，并且通常就是这样干的。图 B.2 显示了在本书

写作时微软主页的样子（注意这个页面是按我的口味定制的，你的页面看起来可能有一点不同）。

Web 链接	你可以使用微软位于 http://www.microsoft.com/Search/Default.asp 的特殊搜索引擎简单快速地搜索其 Web 站点，该搜索引擎提供了特别的功能，使你能够查看单独一类微软产品或者查看整个站点，你还可以选择搜索类型来执行搜索（搜索类型包括了传统的逻辑搜索）。

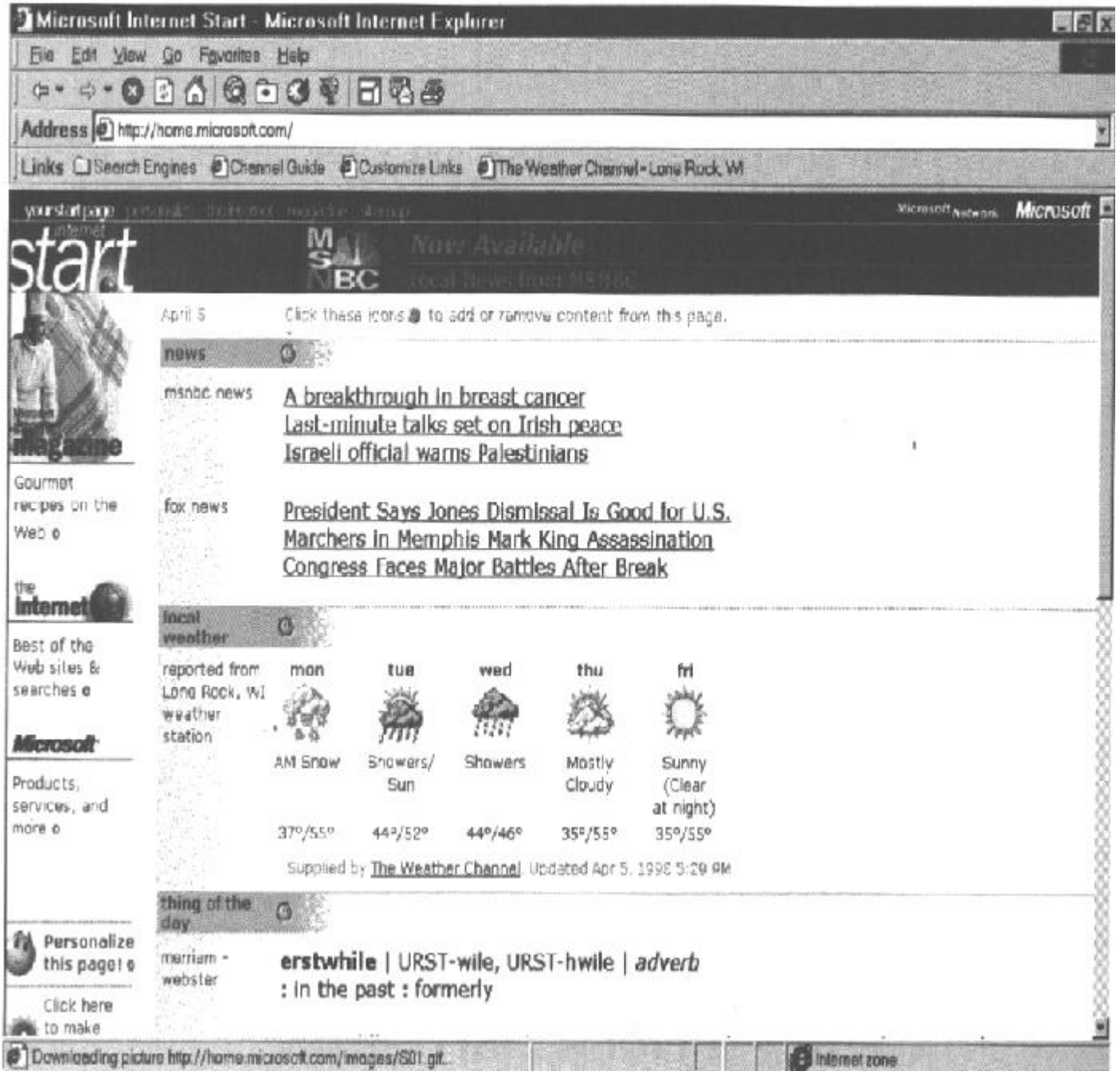


图 B.2　**微软的** Web **主页提供给你一些如何快速查找信息的线索**

因此，你需要回答的问题是不求助于本附录早先讨论的搜索技巧来查找信息的方法。微软通常用 Web 页面的同样模式来设计搜索，因此与一种搜索方法保持一致是相当容易的。

看看页面的左边，你将看到一栏信息，这栏信息通常包含相关信息的列表。单击表中的一个链接，将带你到微软 Web 站点的另一部分——在这种情况下是一个更一般的区域。

你还会发现在大多数微软 Web 页面上获得确切信息的方法，在这种情况下，你需要用下拉列表框，其中包含一个选择地点框，然后单击 Go，这个页面选择的站点能带你到提供不同语言的地方，而其它 Web 页面将带你到这样的地方，在那里你能找到有关你正在使用的语言的明确信息，以及其它一些感兴趣的主题。

寻找新闻组

获取信息不是使用 Internet 的唯一理由，大量新闻组的存在允许你与其它人交换信息，他们还让你获取额外信息或者学习如何完成一项工作。

技巧 一定要查看一下本书中所有的关于新闻组方面的 Web 链接，本书中差不多每章都包括一个这样的链接，能帮助你找到 Web 站点或新闻组（有时候两者均可以找到）。

微软提供了许多新闻组，如果没有某种类型的菜单系统的话，要找到合适的新闻组可能会有一点困难。弄清你对新闻组的需求分为三步。首先，去

http://support. microsoft.com/support/news/default.asp **站点，你将发现如图** B.3 **所示的页面，如你所见，显示这个页面的目的就是让你选择一个产品类型。**

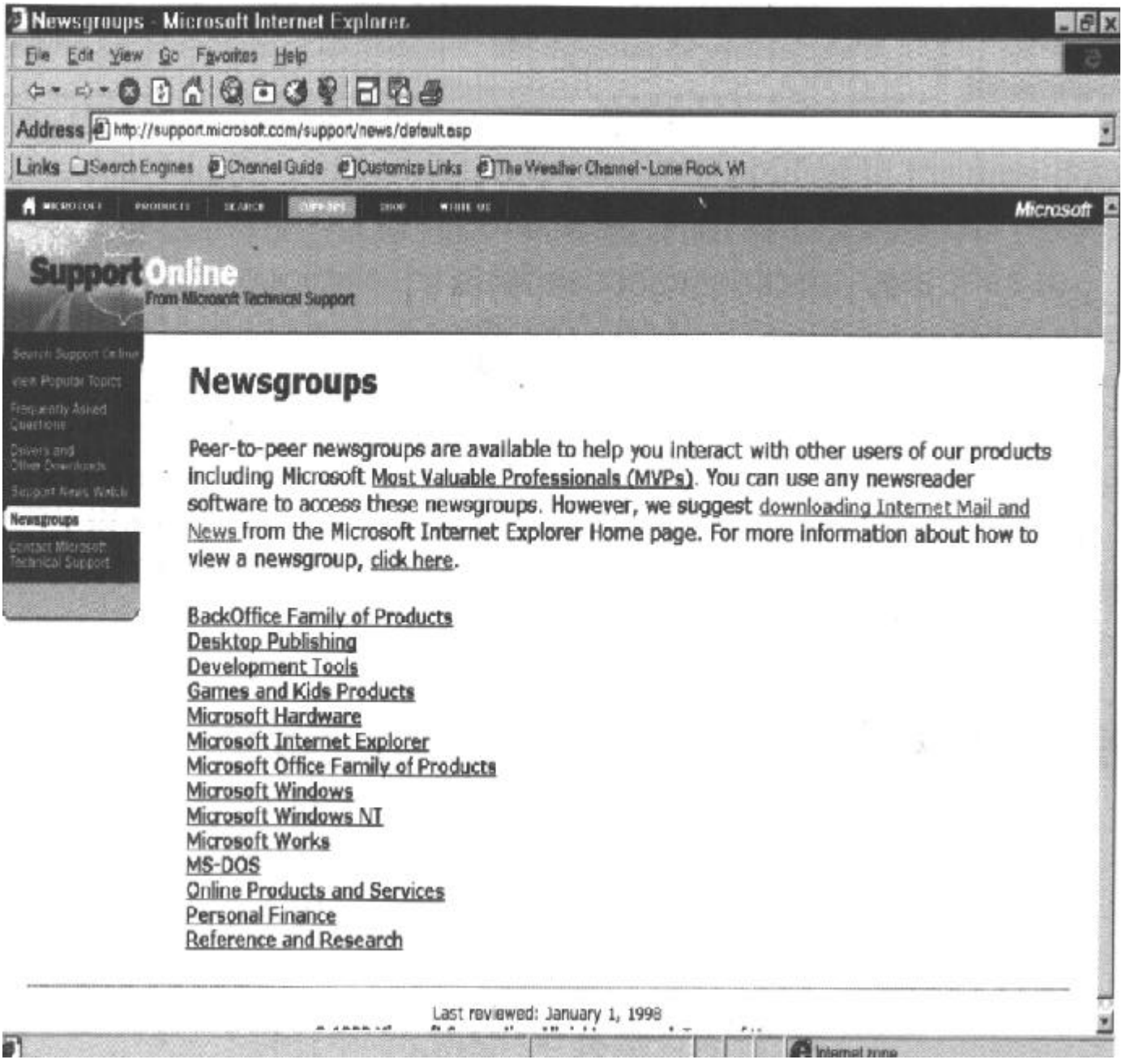


图 B.3 **找到合适新闻组的第一步是选择一个产品类型**

由于大多数读者用 Windows NT 作为开发平台，因此下面我将选择 Windows NT 项。在编著本书时，我还需要选择 Windows NT 的特定版本——对本例来说，我将选择 Windows NT Server and Workstation 链路。图 B.4 显示了你将看到的景像：在新闻组中你可以找到的一系列 Windows NT 信息。

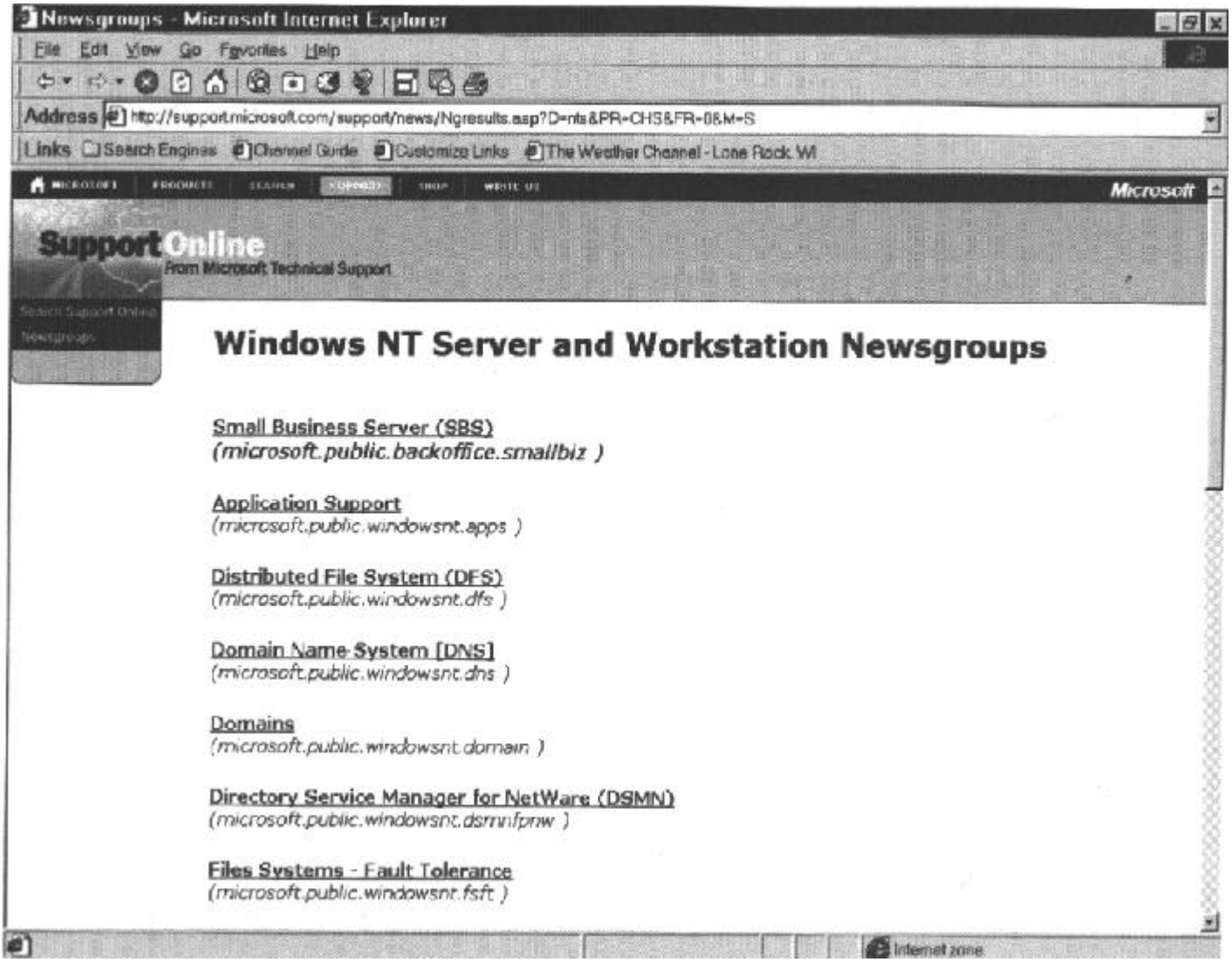


图 B.4　一旦选择了产品类型，你将发现关于该类型产品有些什么样的新闻组是可用

一旦找到了需要的信息类型，单击到该新闻组的链接，像 Outlook Express 这样的新闻阅读器会自动弹出。注意新闻阅读器中的订阅复选框并不自动屏蔽，如果你发现新闻组提供的信息非常有用，并且想自动更新你的新闻阅读器，那么你需要选中这个复选框。

B.3 微软知识库

在全面深入地查看微软在 Internet 上提供的东西之前，我想搜寻一下叫做微软知识库（Microsoft Knowledge Base）的特殊微软站点。这个站点不像你在 Internet 上见到的其它站点那样标准，实际上它是一个集文章、白皮书及其它应用微软产品所需要的信息资源于一体的库，我们将随后讨论这个特殊论坛是如何将这么多东西塞在一起的。现在，我将说明一下这个站点为什么是重要的，以及如何使用它。我经常到微软知识库查看关于微软产品未来发展方向的信息，例如，我曾找到一份白皮书，这份白皮书是关于“Windows 95 中的程序管理器主题”的。你也能找到人们常常遇到的大量问题——这些人就像你一样——以及微软建议的解决方法。有一份白皮书告诉我关于“Windows 95 及 After Dark2.0 主题”的情况。这是我曾期望得到的。最后，你可以找到关于某个产品的技术细节，我曾经找到一份白皮书，它提供了“Windows 95 中系统资源的解释”，这是在当我浏览知识库时找到的。很容易想到知识库更像一个传真支持热线，而不是别的什么，然而，你用来与其联系的方法是如此的不同。

大多数情况下，应用动态支持的问题是，除非你知道你在找什么，否则你

很可能找不到所要的信息。我曾接通好几条传真支持热线，他们很不友好，以至于除非你知道文章的名字或其代号，否则会一无所获。微软知识库则不同，它提供一个搜索引擎，你可以用来实际查找所要找的信息。

那么，你如何用这个时髦的微软产品呢？去 http://support.microsoft.com/support/ c.asp?M=F 看看。当你到达那儿时，你将看到如图 B.5 所示的搜索页面。

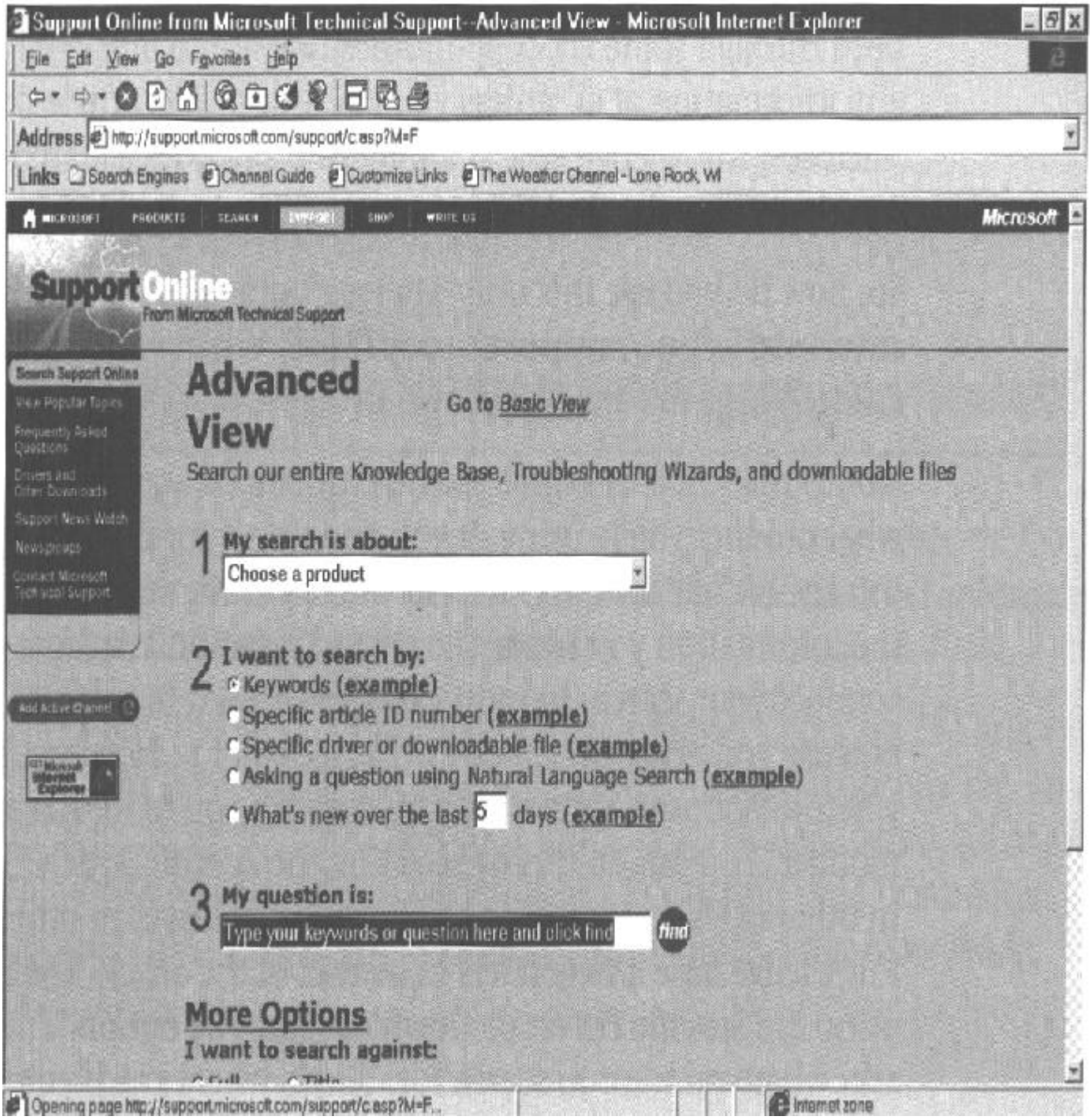


图 B.5　软知识库提供易于使用的搜索引擎来帮助你找到需要的信息

注意，你需要回答三个基本问题，首先是你正使用的产品是什么，如果在下拉式列表中未见到你所用的产品，那么微软知识库很可能不提供你所需要的信息。第二个问题是问你想如何指导你的搜索，多数情况下你会执行关键词搜索，因为你实际上还不知道寻找什么。然而，许多时候会有人请你查看指定文章的代号，这个人是微软支持的。你可通过选择指明文章代号选项来达此目的。在另一些时候，你可能想下载一个驱动程序或其它文件，这些程序或文件是由知识库中的文章引用的，而非下载文章本身。要做到这点，可用指明驱动程序或下载文件选项来实现。第三个问题是你想搜索什么。此处你将输入关键词、文章代号、或你想找的文件名，一旦你回答了这三个问题，就可单击查找（Find）来搜索知识库了。

这个页面确实提供了两种增强功能，可以用来搜索结果（第一个选择在图 B.5 中可以见到，应当在这个搜索页面上花些时间以熟悉这些选项），第一个选项让你选择执行全文检索（所花时间较长，但提供的信息全面），或者只是执行主题检索。第二个选项允许你将几个不同文章的摘要加到搜索页面结果的主题下显示出来，这些摘要有助于你找到所需的信息，而不用实际去读这篇文章，经常是文章前几行将告诉你这篇文章是否为你寻找的文章。

词 汇 表

本词汇表有几个重要的特征必须清楚。首先，整本书中每个缩写词都列在这里，即使你很可能已经知道这个缩写词的意思。这样做的目的是为了正确使用本书，你总可以找到所需的东西，这是毫无疑问的。其次你需要知道的是这些定义只适用于本书，换句话说，当你查这个词汇表时，你所见到的词是在使用它们的上下文中定义的，这可能与当前整个业界的用法一致，也可能不一致，因为计算机工业是如此频繁地改变词的含义。最后，大多数情况下，这里定义是以谈话的语气下的，这意味着为了理解方便，会牺牲一点准确性。

Web 链接 如果在你刚买的计算机字典中找不到你需要的缩写词会怎么样呢？幸运的是，在 Internet 上有两个站点可以给你提供帮助。第一个是 Hill Associates 公司，其站点为 http://mailme.hill.com/acronyms/。这个站点更新相当频繁，且只提供缩写词。第二个站点是 BABEL：A Glossary of Computer Oriented Abbreviations and Acronyms（面向简写及缩写词的计算机词汇表）。你可在 http://www.access.digex.net/%7Eikind/babel.html 处找到它，尽管这个站点不像前一个更新的快，但它却同时提供缩写词及简短的定义，这是它的优势所在。

128 bit/40 bit encryption Key	128 位/40 位加密密钥，见 Encryption level
Access control entry	访问控制项，参见 ACE
Access control list	访问控制列表，参见 ACL
ACE(Access control entry)	访问控制项。Windows NT 专用的安全组件。 每个对象（包括从一个文件到一段内存的所有东西）包括两个访问控制列表（ACL），这些列表决定了用户、系统单元、或者其它类型的对象对该对象拥有什么样的访问类型。在每个 ACL 内是一个或多个访问控制项（ACE）。对每个能够访问该对象的用户、组、或者其它对象都拥有一个项。这个项定义了什么样的访问被授权，例如，一个文件对象可授权读和写。
ACL(Access control list)	访问控制列表。 Windows NT 专用的安全组件。 有两种类型的访问控制列表：其一是安全访问控制列表（SACL），它控制了 Windows 的审计功能，另一类为自由访问控制列表（DACL），它控制那些实际使用对象的程序，访问控制列表包含一个或多个控制项，这决定了 ACL 允许的给予每个用户或对象的实际访问权限。
ActiveX Control	ActiveX 控件，参见 OCX

续表

ActiveX Data Object(ADO)	ActiveX 数据对象，基于 OLE-DB 建立连接的局部和远程数据库访问技术。ADO 是一套封装函数，它使得使用 OLE-DB 和基本 OLE-DB 提供者更容易。ADO 被设计用来替换 DAO，并作为 ODBC 的一个附加部分。
ActiveX Document	ActiveX 文档。用在 Internet 上的基于 COM 的技术之一，用来以 Internet 内部不支持的格式显示文档，例如 Word for Windows DOC 文件格式。利用 ActiveX 文档，OLE 服务器可以接管浏览器的框架（菜单及像滚动条等其它特征）。并且在浏览器窗口显示文档。ActiveX 文档亦被叫做活动文档（ActiveDocument）。
ActiveX Movie	ActiveX 电影。支持 COM 的几种技术之一，用于在 Internet 上通过使用诸如 AVI 文件等特殊文件格式来播放实时视频和声音。ActiveX 电影甚至允许公司在特定站点不需要人员参与的情况下提供联机展示。
ADO	参见 ActiveX 数据对象
Animated GIF	动画 GIF，参见 GIF

续表

API(Application programming interface)	应用程序接口。一种定义一组标准函数调用和其它接口元素的方法。它通常定义高级语言与低级元素之间的接口，这些低级元素被设备驱动程序或操作系统所使用。其最终目标是给应用程序提供访问操作系统或设备驱动程序特性的某种类型的服务。
Application programming interface	应用程序接口，参见 API
ASP	活动服务器页面
ALT	活动模板库
AVI (audiovisual interface)File Format	AVI（音频视频）文件格式。一种以数字形式同时包含声音和图像的特殊文件格式，AVI 是当前在 Internet 上传输多媒体文件的最流行方法。
Binary values	二进值制。指 Windows 注册中基数为２的数据表示法，通常用来存放状态标志或者其它以二进制格式存放的信息。

续表

BLOB（Binary large object）	大的二进制对象。数据库表中的一个特殊字段，它接收像位图、声音或文本等对象作为输入。该字段常和 DBMS（数据库管理系统）的 OLE 能力联系在一起， 但是有些第三方产品使得在老的数据库文件格式如 Xbase 的 DBF 文件格式中加入 BLOB 支持成为可能，BLOB 字段总是隐含 DBMS 的 OLE 客户支持。
BMP file	BMP 文件。Windows 的标准位图数据格式，这是种光栅图形数据格式， 不包括任何形式的压缩。它通常由 Windows 使用，但 OS/2（以及其它不同操作系统）也能用这种数据格式来存放各种类型的图形。
Browse	一种特殊应用接口元素，设计用来为用户显示数据库或者其它存储媒体（例如由某些图形应用提供的略图框架）的大概情况。将浏览想做其余存储区域的内容表，浏览通常包括几个数据存储单元（大多数为记录或者图片略图）的部分内容。用户可以转而看看整个存储单元。一个浏览格式通常包含滚动条或者其它高速界面元素，使得用户可以方便地从存储媒体的一部分转移到另一部分。

续表

Browser	浏览器。一种特殊的应用程序，通常用来显示从 Internet 上下载的数据，最通用的 Internet 数据形式是 HTML（超文本标记语言）页面。然而，现代浏览器还能显示不同类型的图形，甚至能直接显示标准桌面应用程序文件，如 Word for Windows 文档。浏览器的实际功能千差万别，这取决于软件厂商和所用的工作平台。
CAB(cabinet) file	CAB 文件。类似于 ZIP 文件的压缩格式文件，用来从某个地方向另一地方传输代码和数据，CAB 格式通常仅限于开发人员使用。
CCITT(Consultative Committee for International Telegraph and Telephony)	国际电报电话顾问委员会。该组织现在叫 ITU（国际电话联盟）。详见 ITU。

续表

CGI(Common Gateway Interface)	通用网关接口。在 Internet 上从客户机向 Web 服务器传输数据的通用方法之一。CGI 依赖于定义应如何解释数据的脚本，有两种基本的数据传输类型，用户可以向服务器传输新的信息，或者查询服务器上已存在的信息。例如，要求用户输入姓名及地址的数据表格就是第一类传输的例子。Internet 上的搜索引擎页面（帮助用户查找其它站点信息的页面）是第二类传输的例子。一旦 CGI 脚本完成后，Web 服务器通常用传输一页新的信息来提供某种类型的反馈。这可以很简单，就像声明一个数据项或者为一个数据项声明一个 Internet 站点的列表那样。
Class ID	类标识。参见 CLSID
Client	客户。从文件或者其它服务接收数据、服务、或者资源的接收者。这个术语指的可以是一个工作站，也可以是一个应用程序。服务器可以是另一台 PC 机或者另一个应用程序。

续表

CLSID(Class ID or identifier)	类标识。在注册表中给每个对象分配一个唯一标识的方法。也指各种高级语言结构，每个对象必须提供唯一的类标识（CLSID），标识是由生成该对象的机器在本地产生的，这个对象用某类特殊软件生成（例如，微软的 OLE 2 SDK 提供一个生成标识的工具）。像 Visual Basic 和多数 C 编译器等高级语言通常为编程人员自动执行 CLSID 生成序列。
COM(Component Object Model)	组件对象模型。微软制定的面向对象代码与数据封装方法及转移技术。它是诸如 OLE（对象链接及嵌入）和 ActiveX（OCX 的另一个名称，是一种面向对象的代码库技术）等技术的基础。COM 局限于本地连接，DCOM （分布式组件对象模型）是允许在 Internet 环境中传输数据和使用 OCX 的技术。微软的最新技术是 COM+， 该技术将首先出现在 Windows NT 5 中，其主要目的在于将当前 COM 提供的功能扩展到像代码的远程执行这样的新领域。
Common Gateway Interface	通用网关接口，参见 CGI
Component Object Model	组件对象模型，参见 COM

续表

Container	容器。面向对象的一个术语，它已成为 OLE 的一部分， 容器是用于保存对象的驱动器、文件或者其它资源。容器通常作为对象本身来引用。
Cookie	Internet 浏览器用来存储指定站点设置或者其它指定 Web 页面信息的一个或者多个特殊文件，该文件的目的是存储一个或者多个变量的值，以便用户下次访问一个站点时 Web 页面可以恢复它们。Web 页面的主人使用如诸如 JavaScript，Java，VBScript 或 CGI 编程语言编写页面时，总是保存和恢复这些 Cookie。 多数情况下，这是 Web 页主能访问客户站点硬盘的唯一文件。这些 Cookie 可以以一个或者多个文件的形式存在于硬盘的任何地方，这取决于当前所用的浏览器。微软的 IE 为每个站点使用一个文件存储 Cookie，并将它们放到 Cookies 文件夹中，该文件夹通常位于 Windows 主目录下，Netscape Navigator 使用叫做 COOKIE.TXT 的单个文件来存储所有站点的 Cookie，该文件通常存放在 Navigator 主文件夹里。
CryptoAPI	加密应用编程接口，参看 Cryptographic Application Programming Interface。

续表

Cryptographic Application Programming Interface （CryptoAPI）	加密应用编程接口。微软提供的使得软件开发商能在其应用程序中加上加密技术的方案。该方案使用 128 位加密技术，这意味着开发商不能将其产品出口到美国和加拿大以外的地方。
Cryptographic Service Provider	加密服务提供商，参见 CSP。
CSP(Cryptographic Service Provider)	密码服务提供商。在 Internet 上验证公司、开发商及个人身份的特殊公司， 这种身份检查允许公司发出一个电子证件，可以用来实现秘密传输，在一个具体的组群中通常提供几个级别的身份证，例如，有三种级别的个人身分，最低一等通过 Internet 邮件地址来验证个人身分，最高一等要求使用验证身份的语句提供书面证明。当你进入一个验证的站点或者试图下载像 ActiveX 控件之类验证的文档时，浏览器会在屏幕上显示电子证件，允许你根据实际情况作出安全决定。
CSS	层叠式工作表

续表

DACL（Discretionary Access Control List）	自由访问控制列表。Windows NT 专用的安全组件，DACL 控制谁能实际使用该对象。你可以为一特定对象分配群用户或者个人用户。
DAO	数据访问对象，参见 Data Access Object
DASS	分布式安全认证服务
Data Access Object(DAO)	数据访问对象。微软引入的一种老的数据访问技术，它依赖于微软的 Access JET 引擎来访问本地数据。DAO 不提供远程访问功能， 尽管有些编程人员能用它建立不可靠的连接，ADO 和 OLE-DB 已很大程度上代替了这种技术。
Database Management System	数据库管理系统，参见 DBMS。
DBCS	双字节字符集。

续表

DBMS（Database Management System）	数据库管理系统。表、表单、查询、报表及其它数据元素的集合，它作为数据处理的中心，可由一个或者多个用户访问。大多数 DBMS（除了那些自由表格或者基于文件的 DBMS 外）依赖于一系列表来存储信息。每个表中包含记录（行），每个记录由独立的数据域（列）组成。常用的 DBMS 包括 Access、 Paradox、dBASE 和 FileMaker Pro。
DCOM（Distributed Component Object Model）	分布式组件对象模型。Internet 上使用的组件对象模型（COM）的高级形式。这个独特的格式允许通过 Internet 或者其它非本地资源传递数据，它增加了执行异步传输和同步传输数据的功能，这就防止了客户应用程序在其等待服务器应答时被阻塞。更多的细节请参见 COM。

续表

DDE（Dynamic Data Exchange）	动态数据交换。 从一个应用程序剪切数据并粘贴到另一个应用程序的能力。例如，你可以剪切一个画图程序产生的图像并将其粘贴到字处理文档中。一旦粘贴后，原始应用程序对数据所做的改变并不反映到粘贴的数据中，DDE 还提供一种与应用程序通信的方法， 该应用程序支持 DDE 并且要求数据， 例如， 可以用 Excel 宏来调用微软 Word 并且请求某个文档文件的内容。有些应用程序也使用 DDE 来实现文件相关策略，例如，当用户需要打开或者打印文件时，微软的 Word 软件用 DDE 来代替命令行开关以增加灵活性。
DDF （ Diamond Directive File）	类似于 INF（信息）或者 BAT （批处理）文件，DDF 为诸如 DIANTZ 的 CAB 生成实用程序提供把一个或者多个文件压缩成一个存储文件的指令。CAB 文件通常用来通过 CD-ROM 或者其它相类似的媒体在本地发布数据， 或者通过 Internet 或其它服务器连接远程地分发数据。DDF 还能列出完整安装所需的各种文件，这些文件可以保存在其它地方。正常情况下， 这些遗漏的文件已经安装在用户的计算机中，因此， 再次下载他们只会浪费时间，DDF 使得仅当需要时才下载它们。

续表

DDK	设备驱动程序开发软件包
Diamond Directive File	参见 DDF
Digital Signatures Initiative	数字签名协议，参见 DSI
Discretionary Access Control List	自由访问控制列表，参见 DACL
Distributed Component Object Model	分布式组件对象模型，参见 DCOM
DLL（Dynamic Link Library）	动态链接库。根据需要载入内存的特定形式的应用程序代码。 它不能自己单独执行，一个 DLL 包含一个或者多个具体的例程， 这些例程由应用程序使用，以提供不同的功能。例如，DLL 可以提供一组用于在硬盘上访问信息的通用文件对话框。可以有多个应用程序使用 DLL 提供的功能，这在多个程序同时运行时可以减少过量的内存需求。
DNA	分布式 interNet 应用程序

续表

Drag and Drop	拖放操作。在面向对象操作系统不使用常规方法打开文件的一种技巧，例如，这个系统允许用户抓住某个文档文件，将其拖至打印机上，然后放下它，打印机将按其缺省设置打印这个文档。
DSI（Digital Signatures Initiative）	数字签名协议。由 W3C（万维网财团）发起的克服通道级安全某些限制的标准。例如，通道级安全不能处理文档及应用程序的语义，通道也不能非常有效地使用 Internet 的带宽，原因在于所有的处理都发生在 Internet 上而不是客户机或者服务器上。这个标准定义了传输签名的数学方法——这个签名本质上是某个指定个人或公司的唯一代表。DSI 还提供标出安全属性（DICS2）的新方法，以及确认的新格式（PEP），这个标准还建立在 PKCS#7 和 X509.V3 标准之上。
Dynamic Data Exchange	动态数据交换，参见 DDE
Dynamic Link Library	动态链接库，参见 DLL
EIT SHTTP	扩展 Internet 标识保密超文体传输协议
Encryption	加密，参见 Cryptographic Application Programming Interface

续表

Encryption Level	加密级别。对文件的加密强度，通常，加密密钥的大小是加密强度和水平的决定因素。多数 Internet 浏览器和本地应用程序使用两种大小的密钥：40 位和 128 位。40 位密钥可提供达 240 个密钥组合，被认为是比较难以破译的，128 位密钥可提供达 2128 个密钥组合，被认为是非常难以破译的。当前只有 40 位密钥技术是被美国政府批准出口到美国或者加拿大以外地区的。额外信息请参见 Cryptographic Application Programming Interface。
File Transfer Management System	文件传输管理系统，参见 FTMS
File Transfer Protocol	文件传输协议，参见 FTP

续表

FTMS(File Transfer Management System)	文件传输管理系统。Proginet 公司引进了这项 ActiveX 技术，它将主机上的数据带到了桌面，其 Fusion FTMS 可以同任何支持 OLE 容器的开发语言一同工作，如 Delphi，Visual C++和 Power Builder。本质上，你可以将一个 ActiveX 控件放到窗体上，定义到哪找数据，然后使用该控件来建立连接。利用这个控件减少了实现和维持主机连接所需的工作量。某个主机上的特殊传输服务器完成打包工作，通过自动方式实现所有传输要求。在客户能够访问之前，操作者不再需要到公司的 Web 站点手工下载所需文件，用户可以直接访问主机上的数据，将其下载到本地硬盘上。
FTP（File Transfer Protocol）	文件传输协议。Internet 的几个通用数据传输协议之一，这个协议专门处理以文件下载形式传输数据。用户可见到以目录列表列出的可用文件列表，FTP 站点可以选择 DOS 或 UNIX 格式列出文件，尽管 DOS 格式相当少。与 HTTP 站点不同，FTP 站点提供了有限信息层次，通过使用目录和子目录，很像大多数工作站硬盘上的文件目录结构。
GDI	图形设备接口

续表

GIF(Graphics Interchange Format)	图形交换格式。用来在 Internet 上传输数据的标准文件格式，这种文件格式有几种不同的标准——最新标准是 GIF89a，它在大多数 Internet 站点都在使用。 GIF 标准最初由 CompuServe 公司引进，其目的是为了减少下载图形所需的传输时间、并降低任何一位传输错误所造成的影响。另一种形式的 GIF 是动画 GIF。它允许开发商在一个文件内存储几幅图像。每个文件中都有一个或者多个确定块边界、下一幅图像相对于显示区域的显示位置、以及其它显示特性的控制块。浏览器或其它专门设计的应用程序将按照顺序一次显示一幅图形画面，这样就产生了动画效果。
Gopher	几种通用 Internet 数据传输协议之一。与 FTP 相似，Gopher 专门处理文件传输，然而，两个协议不同之处在于 Gopher 总是用 Unix 文件命名方式，并且提供比 FTP 更友好的界面。尽管 Gopher 传输比 FTP 提供的传输更可靠，但 FTP 站点远比 Gopher 站点多。
Graphics Interchange Format	图形交换格式，参见 GIF
GSS-API	通用安全服务应用编程接口
GUID	全局唯一标识

续表

HTML（Hypertext Markup Language）	超文本标记语言。Internet 的脚本语言，它依赖于使用标记（尖括号〈〉内的关键词）以与平台无关方式在屏幕上显示格式化的信息。这个脚本语言的与平台无关特性使得执行某些基本任务相当困难，比如在一指定的地方放置一个屏幕元素。然而，该语言确实提供了在屏幕上使用字体、颜色和各种其它增强功能，该语言还有显示图形画面的标记。虽然不是所有的浏览器都支持这些新加标记，然而最近还是增加了一些诸如 VBScript 和 JavaScript 这样的更为复杂的脚本语言使用的标记。最新增加的标记允许使用 ActiveX 控件。
HTTP（Hypertext Transfer Protocol）	超文本传输协议。Internet 的几个公共数据传输协议之一，这个协议专用于屏幕信息显示，如数据项形式或信息的显示。尽管可以使用 HTTP 来显示非格式化文本，但 HTTP 依赖于描述特殊屏幕显示元素的脚本语言 HTML。
Hypertext Markup Language	超文本标识语言，参见 HTML
Hypertext Transfer Protocol	超文本传输协议，参见 HTTP

续表

IDAPI（Independent Database Application Programming Interface）	独立数据库应用编程接口。以 Borland 为主的一些公司引进的一套 Windows 函数调用和其它接口元素。IDAPI 设计用来通过使用公共接口和与数据无关的访问方法，改善访问数据库文件中的信息。
IETF（Internet Engineering Task Force）	Internet 工程任务组。专门解决 Internet 上技术问题的标准化组织。该组织能够批准各种标准，不论这个标准是在该组织内部产生的，还是由其组织外其它群体的努力产生的。例如，微软已通过该组织批准了几个新的 Internet 技术。如果技术标准被批准后，这些技术将会成为 Internet 方面完成数据传输或其它特殊任务的标准。
IIS	Internet 信息服务器
IMTF（Internet Management Task Force）	Internet 管理任务组。负责在 Internet 上实现新技术的标准化组织。问题是该组织主要由志愿者组成，与其它标准化组织相似，IMTF 制定标准的步伐也十分缓慢。事实上，由于它的步伐如此之慢，许多公司提出了他们自己的解决办法，以使得 Internet 成为更友好的工作地点。例如，微软开发了 ActiveX 技术来响应特定的与 Internet 相关的问题。而 Netscape 开发了 Netscape ONE（开放网络环境）。

续表

Independent Database Application Programming Interface	独立应用编程接口，参见 IDAPI。
INF（information）File	INF（信息）文件。设备或应用程序配置的一种特殊表格，它包含了安装或者配置设备或者应用程序时 Windows 所需的所有参数。例如，一个应用程序的 INF 文件可能包括数据文件的位置和 DLL 之间的依赖关系。应用程序和设备的 INF 文件都包含使 Windows 能够识别应用程序和设备的注册项和 INF 文件项。
International Telephony Union	国际电话联盟，参见 ITU
Internet Engineering Task Force	Internet 工程任务组，参见 IETF
Internet Management Task Force	Internet 管理任务组，参见 IMTF

续表

Internet Protocol	Internet 协议，参见 IP
Internet Server Application Programming Interface	Internet 服务器应用程序编程接口，参见 ISAPI
IP（Internet Protocol）	Internet 协议。Internet 使用的 TCP/IP 协议的信息交换部分，IP 是实际使用的数据传输协议，它定义了信息如何打包，以及如何从一地传到另一地。TCP （传输控制协议）定义了如何实际实现数据传输。标准化组织宣称现在 IP 的问题之一是它不对数据包加密——任何人都可以阅读 Internet 上的数据包。IP 的未来版本将使用某种形式的加密技术来达到数据包加密的目的。同时，有些公司已经将 TCP 与其它技术结合了起来， 使其提供短期的加密技术。
IPSec	Internet 协议安全协议
ISA	ISAPI 服务器应用程序，参见 ISAPI Server Application

续表

ISAPI(Internet Server Application Programming Interface)	Internet 服务器应用编程接口。一组使微软 Internet 信息服务器（IIS）及相关产品（比如 Peer Web Server）更容易使用的函数调用和接口元素。本质上，这套 API 调用使得编程人员能够访问服务器本身。这种访问通过一系列 ActiveX 控件而非使用脚本语言使得全面访问 Internet 服务器功能变得更加容易。
ISAPI Server Application（ISA）	ISAPI 服务器应用程序（ISA）。放置在 Web 服务器上以 DLL 形式表现的一种特殊格式的 ActiveX 控件。有几种方式调用这个 DLL，比如客户端来了服务请求或者发生了某个特殊事件。有两种形式的 ISA：过滤器及扩展。过滤器允许你通过监视服务器上的事件保留或者排除某些东西。例如，你可以生成一个 ISA 过滤器，使得只有输入正确的通行字才能进入你的 Web 站点。另一类型的过滤器是能够防止一定大小的文件上载到你的 Web 站点的 FTP 服务器上。扩展更像应用程序或者后台进程，例如，你可以生成一个扩展，允许用户与一个数据库交互作用而不用借助于脚本。同样的扩展也可以根据用户输入和你的服务器上数据库的内容动态生成 Web 页面。

续表

ITU（International Telephony Union）	国际电话联盟。CCITT 为其前身，该组织以其与调制解调器通信相关的标准而闻名于世。然而，近些年来，它也已经开始制定传真和 Internet 标准。所有老的 ITU 标准依然使用 CCITT 标记。新标准使用 ITU 标记。与其它标准化组织不同，ITU 是个多国组织， 其工作人员是来自许多不同国家的代表。
JEPI	联合电子支付协议
Joint Pictures Entertainment Group File Format	参见 JPEG 文件格式
JPEG File	JPEG 文件，参见 JPEG File Format
JPEG （Joint Pictures Entertainment Group ） File Format	Internet 上使用的两种图形文件格式之一。这是一个向量文件格式，用来提供高分辨率影像或者图片。

续表

LAN（Local area network）	局域网。用于把一组 PC 机彼此连接起来、并且和/或与小型机或者主机连接起来的硬件和软件的组合。实际使用中有两种主要的网络模型：即对等模型和客户机/服务器模型。对等模型不要求提供服务器。另外，同一组中的所有工作站可以共享资源。客户机/服务器模型使用中央服务器作为资源共享。然而，它也提供了以有限方式使用本地资源的几种特殊方法。
Local Area Network	局域网，参见 LAN
Locally Unique Identifier	本地唯一标识，参见 LUID
LUID（Locally Unique Identifier）	本地唯一标识。实际上就是个指向对象的指针，为安全起见，LUID 标识每个过程和资源。换句话说，即使一个用户有两个一模一样的资源拷贝（比如一份文档），这两个拷贝都各自有唯一的 LUID。这种标识方法防止了 Windows NT 下某种类型的安全访问冲突。
Macro	宏。一种在磁盘文件或者在当前文档内记录敲键及其它与编程相关任务的编程形式。许多应用程序提供宏记录器，记录所敲的键和按下的鼠标，这意味着在绝大多数情况下，你甚至不用编写宏。宏在电子表格中特别流行。绝大多数宏使用某种形式的 DDE 来完成与 OLE 相关的任务。

续表

Mail Handling Service	邮件处理服务，参见 MHS
Mail Transfer System	邮件传输系统，参见 MTS
MCI	媒体控制接口
MFC（Microsoft Foundation Classes）File	微软基础类库文件。使许多微软应用程序正常运行所需的 DLL 集合。这些文件包含了作为建立应用程序基础的共享类。例如，按钮就是这些文件中一个单独的类。正常情况下，MFC 文件存放在 Windows 的 SYSTEM 文件夹中——这些文件使用 MFC 作为文件名的开始字母。
MHS（Mail Handling Service）	邮件处理服务。一种对用户邮件加密、解密及执行其它邮件管理服务的方法。许多 NOS（网络操作系统）提供某种类型的 MHS 作为基本系统的一部分。在 Internet 上有好几个标准可用于提供作为 Web 站点一部分的 MHS，其中两个最引人注目的规范是 IETF 的 IETF RFC1421 和 ITU（前身为 CCITU）的 X.400。
Microsoft Foundation Classes File	微软基础类库文件，参见 MFC File

续表

MAT	消息传输代理
MTS（Mail Transfer System）	邮件传输系统。一种从一地向另一地传递邮件的方法。绝大多数情况下，这种方法要求某种形式的加密以及其它与传输有关的技术。多数网络操作系统（NOS）提供某种类型的 MTS 作为其基本服务的一部分。然而，Internet 要求特殊的传输机构。在 Internet 上有好几个标准可用于提供作为 Web 站点一部分的 MHS，其中两个最引人注目的规范是 IETF 的 IETF RFC1421 和 ITU（前身为 CCITU）的 X.400。
MTS	微软事务服务器
Nested Objects	嵌套对象。以某种方式组装在一起的两个或多个对象，这些对象通常出现在一个容器对象的框架内。嵌套对象允许用多个对象定义为更高高层次对象的属性，它还允许用户把彼此不同类型的对象连接起来。

续表

Netscape ONE（Open Network Enviroment）	Netscape 开放网络环境。基于 Internet Inter-ORB 协议（IIOP ）和公共对象请求代理结构（CORBA）规范的一套特殊的应用程序编程接口和类库，IIOP 和 CORBA 允许编程人员生成定制的 Internet 应用程序。这种定制的好处之一是编程人员不用使用 CGI 或者其它脚本语言就能访问服务器上的数据，而脚本语言是标准 HTTP 所必需的。ONE 现在包括五个基于 Java 的基础类库：用户接口控件（User Interface Control），用户接口服务（User Interface Services），安全（Security），消息（Messaging）和分布式对象（Distributed Object）。将来该规范还计划包括用于数据库、用于 Novell NetWare 目录服务(NDS)的文件服务器目录库访问和其它产品的基础类。这种新技术还要求升级 JavaScript，尚不知道微软是否会支持这种升级。
Network Interface Card	网络接口卡，参见 NIC
Network Operating System	网络操作系统，参见 NOS

续表

NIC（network interface card）	网络接口卡（即网卡）。支持工作站与文件服务器和其它工作站进行通信的设备。它提供了建立连接的物理方法，该卡插在计算机的扩展槽中。连接在卡背后的电缆线构成通信通道。
NOS（Network Operating System）	网络操作系统。运行在文件服务器或者其它以文件和打印共享为中心的设备上的操作系统，这种操作系统除了其它网络实用程序外，通常还提供多用户访问能力和用户计账软件。
Object Conversion	对象转换。把一个应用程序生成的对象的格式和属性转换为另一应用程序使用的对象的格式和属性的方法。该转换把一个应用程序中的数据转移到另一个应用程序中，通常情况下转换过程中不会丢失格式信息，但可以保证从不会损失内容。
Object Linking and embedding	对象链接和嵌入，参见 OLE

续表

Object Linking and embedding database (OLE-DB)	对象链接和嵌入数据库。一种低级的数据库访问技术， 它依赖于 COM 和提供 OLE-DB 提供者的厂商而非 ODBC 使用的 SQL。OLE-DB 既可用于访问远程数据库，也能访问本地数据库。另外，它还能访问不依靠 SQL 的数据管理器。OLE-DB 和 ODBC 是互相合作而不是彼此竞争的数据访问技术，OLE-DB 和 ADO 一起使用，设计用来代替像 RDO 和 DAO 这些老的数据库技术。
OCX （OLE Control eXtension）	OLE 控件扩展。VBX 的一种特殊形式，设计用来使得编程人员在应用程序中更容易增加 OLE 功能。本质上，OCX 是一个添加了编程能力和 OLE 接口的 DLL。
ODBC (Open Database Connectivity)	开放数据库互联。微软引入的一组 Windows 函数调用和其它接口元素。ODBC 设计用来通过使用通用接口和与数据无关的访问方法来访问数据库文件中信息。正常情况下，ODBC 利用 SQL 把客户机 DBMS 专用的命令翻译为通用语言。 服务器上的 ODBC 代理将这些 SQL 请求翻译为服务器专用的命令。
OLE Control eXtension	OLE 控件扩展，参见 OCX

续表

OLE-DB	对象链接与嵌入数据库，参见 Object linking and embedding database
OLE (object linking and embedding)	对象链接与嵌入。将文件名和任何要求的数据打包到一个对象中， 然后将这个对象粘贴到另一应用程序产生的文件中，这个过程叫做 OLE，例如， 你可以将一个图形对象放到一个字处理文档或者电子表格中，当你看到这个对象时，就好象只是简单地将数据从原始应用程序粘贴到当前应用程序中一样（与 DDE 类似）。 一旦建立链接，当你改变原始对象的数据，该对象提供的数据自动改变，一旦嵌入，数据不改变，除非你明确地编辑它，然而数据保持原始形式，你仍需要用原始应用程序来编辑数据。通常通过双击对象你可以启动运行原始应用程序、自动加载所需的数据。
ONE	参见 Netscape ONE（开放网络环境）
Open Database Connectivity	开放数据库互联，参见 ODBC

续表

PCT（Private Communication Technology）	专用通信技术。IETF 和微软正在制定的协议。与 SSL 一样，PCT 用于在客户机与服务器之间低层协议上提供安全通信的方法。它能与任何高层协议（比如 HTTP、FTP 或者 TELNET）一起工作。通过使用加密、验证和数据签名，PCT 设计用来阻止在客户机和服务器之间的通信被窥取。与 SSL 一样，客户验证是可选的。PCT 还假定你已经具备了 TCP 或另外可靠的传输协议。它通过提供扩展的密码协商和其它附加功能，改正了 SSL 中某些延续下来的弱点。
PDF	可携带文档格式。
PEM（Privacy Enhanced Mail）	增强型私人邮件。包括四个已经颁发的 IETF 标准（从 IETF RFC1421 到 IETF RFC1424），这些标准定义了在 Internet 上收发邮件的方法。最重要的技术是以某种方式对邮件加密解密，不用用户过多参与就能确保最佳的个人隐私权，这些标准还涵盖了与邮件加密相关的主题，包括厂商提供服务的验证和 CSP 的使用。
Private Enhanced Mail	增强型私人邮件，参见 PEM。

续表

Private Communicating Technology	私人通信技术，参见 PCT
Private Key File	私有密钥文件，参见 PVK
PVK（Private Key File）	私有密钥文件。保存在客户机或者服务器上的一个文件，用来生成完整的数据加密。当这个文件中的密钥与另一文件提供的公共密钥结合起来时，这个文件成为可访问文件。因为 PVK 文件从来不用从一地向另一地传输， 大大增加了数据通信安全程度。PVK 文件用于各种基于认证的通信类型。例如，从 VeriSign 或者另一组织获得的个人认证将会在你的计算机上产生一个 PVK 文件。开发人员也生成 PVK 文件来使用各种类型的 Internet 技术，比如 ActiveX。生成私有和公共密钥并将它们分配给实际组件的过程叫做签名。同样，通过让人知道文档的作者，可使签名的邮件或者其它通信极大地增强安全性。
PWS	Peer Web Services
RAD（rapid application development）	快速应用开发工具。一种让你设计程序界面并编写代码响应用户在应用程序界面上操作的工具。Visual Basic 和 Dephi 均为 RAD 程序的例子。

续表

Rapid application development	快速应用开发工具，参见 RAD
RDO	远程数据对象
RDS	远程数据服务
Remote Access	远程访问。像使用本地资源那样使用远程资源的能力。在某些情况下，这也意味着下载远程资源来作为本地资源使用。
Remote Procedure Call	远程过程调用，参见 RPC
Rich Text Format	富文本格式，参见 RTF
RPC（Remote Procedure Call）	远程过程调用。像本地那样使用某一机器上的代码和数据的能力。这是非集中式应用程序终将采用的一种高级功能。
RTF（Rich Text Format）	富文本格式。最先由微软引入的格式，它允许应用程序以普通 ACSII 文本存储格式化信息，所有命令以反斜杠开始，例如，\cf 命令通知有 RTF 能力的编辑器在显示一段文本时使用颜色表中的哪种颜色。
SACL（Security Access Control List）	安全访问控制列表。SACL 控制 Windows 的审计功能。用户或者用户组每次访问一个对象并且该对象的审计功能是开放的话，Windows 就在审计文件中产生一个输入项。

续表

SDK	软件开发工具箱
Security Socket Layer	安全套接层，参见 SSL
Security Access Control List	安全访问控制列表，参见 SACL
Security identifier	安全标识，参见 SID
ShellX（Shell extension）	外壳扩展，一种特殊的应用程序，它给操作系统接口增加某种附加能力。绝大多数情况下，应用程序在操作系统能认出它之前必须在注册表中先行注册。
SID（security Identifier）	安全标识。用户访问标记的一部分， 用来标识网络中的用户——它就像账户号码一样。SID 标识的用户标记告知用户属于哪个组，该用户有什么特权。 每个组也有一个 SID，因此用户的 SID 包含了对它所属的组的 SID 的引用。 而不是完全的组访问权限，你通常可以用 Windows NT 下的用户管理工具来改变这个访问标记的内容。
S/MIME	安全/多用途 Internet 邮件扩展

续表

SQL （Structured Query Language）	结构化查询语言。多数 DBMS （数据库管理系统）使用该语言来交换信息，有些还把它用作本地语言。SQL 提供了从 DBMS 获取信息的方法。 它定义了使用哪个或者哪些表、从表中获取什么样的信息以及如何对这些信息进行排序等功能。
SSL（Secure Socket Layer）	安全套接层。最初由 Netscape 提出的 W3C 标准，用来在协议层从客户机到服务器传输加密信息，套接允许以高级协议（比如 HTTP， NNTP 和 FTP）进行低级加密传输。该标准还提供了服务器和客户机验证（尽管客户机站点验证是可选的）。
Stickey Keys	粘连键。由微软提供的几种特殊功能之一，用来帮助更好地使用计算机。这个功能是作为 Windows 95 和 Windows NT 4.0 的可访问小程序的一部分提供的。
Stream object	流对象。用于从一个对象向另一对象传递信息的数据封装容器。例如，流对象可以将数据从应用程序内存移到磁盘上的文件中存储起来。
Structured Query Language	结构化查询语言，参见 SQL。
S/WAN	安全/广域网。

续表

TCP/IP（Transmission Control Protocol/Internet Protocol）	传输控制协议/网际协议。由美国国防部开发的标准通信线路协议，该协议定义了两个设备如何相互通话，可将该协议想像为两个设备使用的某种语言。
Token Calls	标记调用。Windows NT 安全 API 的一部分，负责处理用户访问一个具体的对象。 要获得访问一个对象的权限，所要求的对象必须提供一个标记符号。本质上，一个标记符号是通向安全对象的门票，安全 API 比较由所要求的对象的标记提供的权限与要求获得访问安全对象的权限。如果要求对象的权限等于或者大于那些要求进入的对象的权限，则操作系统授权进入。标记在 Windows NT 下是一致的形式，对用户和外部应用程序不加限制。该操作系统必须使用标记。
UNC（universal naming convention）	（同一命名协议）。一种标识网络资源而不使用具体地址的方法。 多数情况下，这种协议用于驱动器和打印机，但它也用于其它类型的资源。一个 UNC 通常用设备名来替代标识。例如：在远程机器上的一个磁盘驱动器可能称为'\\AUX\DRIVE-C'，使用 UNC 的好处在于，即使资源地址改变了，资源名称也不会改变（当用户在某机器上改变驱动器映射时，这种情况就会发生）。

续表

Uniform resource locator	统一资源定位器，参见 URL
Universal Naming Convention	同一命名协议，参见 UNC
Uri	通用资源标识
URL（uniform resource locator）	统一资源定位器。在 Internet 上标识一个位置的基本方法。资源可以是文件、Web 站点、或者你可以用这个媒体访问的任何其它东西。URL 总是包含三个基本部分，第一部分标识用来访问资源的协议，例如，在某个 URL 开头的字母串 http 总是表示指定的站点使用超文本传输协议并展示某类可视化信息，URL 的第二部分是主机名称，例如：最流行的主机名称为 www，它表示 World Wide Web。URL 的第三部分是一个域，这通常是站点机器的名称和你打算进入的站点类型。（例如，MyCompany.com 会告诉你这个域是一个名叫 MyCompany 的机器以及它是某种类型的商业站点）。站点信息之后是目录，就像在硬盘上的目录一样，因此，像 http://www.MyCompany.com/mysite.html 这样一个 URL 将指向一个使用 HTTP 的 Web 页面，它位于 World Wide Web 上的 MyCompany.com。

续表

VBA（Visual Basic for Application）	应用程序使用 Microsoft Visual Basic 的一种形式。它提供的功能比 VBScript 多，但比 Visual Basic 编程语言的功能少。这个语言的基本原则是不需要过多的学习就达到完全支配整个机器的目的。VBA 最初是设计用来允许用户生成脚本类型的宏并且提供应用程序之间的通信功能。当 VBScript 不能提供足够的能力来完成一项特殊的任务时，VBA 就成为扩展其功能的高级编程语言。
VBX（Visual Basic eXtension）	一种特殊形式的 DLL，包括函数和编程人员接口， VBX 的 DLL 部分要求从应用程序处得到特殊服务，比如打开一个文件；编程人员接口部分作为按钮显示在程序（比如 Visual Basic）的工具条上，单击按钮产生该特殊类型控件的一个实例。
Virtual Reality Modeling Language	虚拟现实建模语言，参见 VRML
Visual Basic eXtension	参见 VBX
Visual Basic for Application	参见 VBA
VPN	虚拟专用网络

续表

VRML（Virtual Reality Modeling Language）	虚拟现实建模语言。一个特殊的脚本（场景描述）语言，允许 Web 站点用少量超前技术传输向量图形影像信息。该语言的价值在于它实际使用非常少的数据来传递所要求的坐标信息。VRML 仍处于实验阶段——传输速率是主要考虑的问题。因为当前拨号连接的带宽比较窄和 VRML 期间发生的改变比较多。用当前的技术，即使用最少的数据传输也不能使 VRML 操作更快。
W3C（World Wide Web Consortium）	一个标准化组织，主要从事 Internet 安全方面的工作，但也参与其它工作，如微软请求的实现 ActiveX 技术的〈OBJECT〉特殊标记。W3C 最初在 1994 年 12 月致力于 SSL（安全套接层）。1995 年 2 月，它专注于 Internet 的应用程序级安全，其当前项目是数字签名协议——该项目是 W3C 于 1996 年 5 月在巴黎提出的。
WAN（wide area network）	广域网。不在一个物理位置、但有机组织起来的两个或者多个局域网（LAN）组成的实体。
Wide area network	广域网，参见 WAN
World Wide Web Consortium	见 W3C
WTS	Web 事务安全